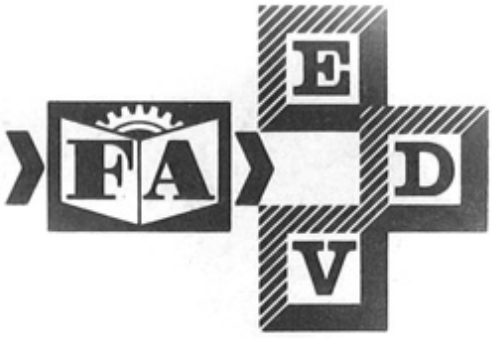
FERNSEHKURS DES DEUTSCHEN FERNSEHFUNKS



Themenkomplex 4/1

Grundlagen der Programmierung

Verlag Die Wirtschaft Berlin

Das Material
entstand unter Verwendung
der Autorenmanuskripte des Fernsehkurses
"Elektronische Datenverarbeitung"

Lektion 15 von Dr. Karl-Heinz Müller, Dresden
Lektion 16 und 17 von Dipl.-Math. Klaus Hentschel, Dresden

die auch die Überarbeitung
für die Veröffentlichung besorgten

Lektor: Dipl.-Hdl. Renate Eyraud

Redaktionsschluß:
10. Juni 1969

Verlag Die Wirtschaft, 1055 Berlin, Am Friedrichshain 22
1969 veröffentlicht · Lizenz-Nr. 122 · Druckgenehmigungs-Nr. 195/169/69
Alle Rechte vorbehalten
Schreibsatz: Verlag Die Wirtschaft Berlin
Gesamtherstellung: Ostseedruck Rostock
ES 21 A · 0,90

Inhaltsverzeichnis

LEKTION 15

Die Problemanalyse

1. Einleitung 4
2. Die Programmierung – eine Etappe der Einsatzvorbereitung 5
3. Die Hauptarbeitsstufen der Vorbereitung 6
4. Die Arbeitsstufen der Problemanalyse 7
5. Das Erarbeiten der Problemanalyse 9

LEKTION 16

Die Programmablaufplanung (I)

1. Einleitung 17
2. Die Darstellung von Programmablaufplänen 17
3. Die Verwendung von Unterprogrammen 21
4. Die Verzweigung eines Programmablaufes 24
5. Die Arten der Programme 28

LEKTION 17

Die Programmablaufplanung (II)

1. Einleitung 29
2. Die zyklischen Programme 30
3. Abschließende Bemerkungen zur Aufstellung von Programmablaufplänen 39
4. Die Zusammenstellung der wesentlichen Sinnbilder für Programmablaufpläne (Kästchenmethode) 41

LEKTION

15

Die Problemanalyse

1. Einleitung

2. Die Programmierung – eine Etappe der Einsatzvorbereitung

3. Die Hauptarbeitsstufen der Vorbereitung

3.1. Die Problemanalyse
3.2. Die Programmierung
3.3. Die Nutzrechnung

4. Die Arbeitsstufen der Problemanalyse

4.1. Die Formulierung der Problemstellung
4.2. Die mathematische Modellierung
4.3. Das Aufstellen von Algorithmen

5. Das Erarbeiten der Problemanalyse

5.1. Die Berechnung von Kurvenscheiben
5.2. Die Lohnberechnung
5.3. Die Umsatzberechnung
5.4. Die Berechnung der Quadratwurzel

1. Einleitung

Im Themenkomplex 4 "Grundlagen der Programmierung" werden die unmittelbar für die Programmierung erforderlichen Vorarbeiten für die Lösung von Aufgaben mit Hilfe einer elektronischen Datenverarbeitungsanlage sowie die Grundzüge der Programmierung behandelt. Dabei werden an einfachen Beispielen die Hauptarbeitsstufen von der Problemanalyse bis zur Programmierung und die Grundlagen der erforderlichen Hilfsmittel, wie die Programmablaufplanung und die Programmiersprachen, erläutert. Den Abschluß dieses Themenkomplexes bilden Hinweise und Anregungen zur Praxis der Programmierung und zur Nutzung von Programmbibliotheken.

In dieser Lektion geben wir einen Überblick über die Hauptarbeitsstufen der Vorbereitung und Durchführung von Aufgaben mit Hilfe einer elektronischen Datenverarbeitungsanlage sowie über die einzelnen Arbeitsstufen der Problemanalyse.

LEKTION

15

2. Die Programmierung – eine Etappe der Einsatzvorbereitung

In Lektion 1 wurde auf die Notwendigkeit der Anwendung der marxistisch-leninistischen Organisationswissenschaft einschließlich Kybernetik, Operationsforschung und elektronischer Datenverarbeitung bei der Gestaltung des entwickelten gesellschaftlichen Systems des Sozialismus eingegangen.
Die umfassende Anwendung der elektronischen Datenverarbeitung gehört zu den Hauptzügen der künftigen Struktur der Wirtschaft der Deutschen Demokratischen Republik. Der Einsatz dieser hochentwickelten Technik bewirkt nicht nur eine organisatorische und strukturelle Veränderung der Arbeitsweise, sondern er verändert den Inhalt der Arbeit überhaupt. Die maschinelle Informationsverarbeitung wird in den kommenden Jahren zu einer alltäglichen, interessanten Arbeit vieler Tausender Werktätiger werden.
In zunehmendem Maße werden nicht nur manuelle Funktionen, sondern auch geistige Arbeiten automatisiert.
Aus dem Themenkomplex 3 "Aufbau und Arbeitsweise einer elektronischen Datenverarbeitungsanlage" wissen wir, daß eine EDV-Anlage zwar über eine hohe interne Verarbeitungsgeschwindigkeit verfügt und über eine Reihe von Ein- und Ausgabegeräten mit der Umwelt korrespondieren kann, daß sie aber nur auf der Grundlage von vorgegebenen Programmen arbeitet.

Die Vorbereitung und Einführung der Datenverarbeitung verlangt ein umfassendes analytisches Durchdringen der Prozesse durch den Menschen. Er muß die Prozesse modellieren, Methoden zur Lösung der gestellten Aufgaben formulieren und der Datenverarbeitungsanlage das Programm vorgeben. Das stellt hohe Anforderungen an die geistige Arbeit, erfordert ein gründliches Durchdenken aller Zusammenhänge und erzieht zum Systemdenken.

Die Arbeiten zur Vorbereitung und Durchführung des Einsatzes elektronischer Datenverarbeitungsanlagen fassen wir unter dem Begriff Einsatzvorbereitung zusammen.

> Die Einsatzvorbereitung der elektronischen Datenverarbeitung ist ein komplizierter, komplexer Prozeß mit vorausschauendem Charakter, der eine zweckgerichtete, wissenschaftsmethodisch begründete, planmäßige, geistig-schöpferische Tätigkeit des Menschen erfordert.

Die Komplexität der Aufgabenstellung und der Umfang der anfallenden Arbeiten machen es notwendig, die Einsatzvorbereitung in Etappen durchzuführen. Folgende Etappen haben sich herausgebildet:

- Studie zur Einsatzvorbereitung,
- Grobsollprojekt,
- Feinsollprojekt,
- Programmierung,
- Umstellen der Organisation und Einführen des Projektes.

LEKTION

15

Im Themenkomplex 4 greifen wir nur ein Teilgebiet der Einsatzvorbereitung heraus. Es geht uns nur um den Inhalt der Programmierung. Inhalt und Umfang der anderen Etappen werden unter Berücksichtigung der Besonderheiten komplexer Aufgabenstellungen, wie sie sich bei der Projektierung von integrierten Datenverarbeitungssystemen ergeben, im II. Teil dieses Fernsehkurses ausführlich behandelt.
Da die Programmierung nicht völlig losgelöst von den übrigen Etappen der Einsatzvorbereitung dargestellt werden kann, wir die Programmablaufplanung und die Programmierung jedoch anhand von Beispielen behandeln wollen, legen wir eine stark vereinfachte Gliederung, wie sie bei der Bearbeitung von wissenschaftlich-technischen Aufgaben üblich ist, zugrunde und unterscheiden folgende Hauptarbeitsstufen:

- die Problemanalyse,
- die Programmierung,
- die Nutzrechnung.

Zum besseren Verständnis werden die ausgewählten Beispiele stark vereinfacht dargestellt, auf die Verflechtungen zu anderen Teilkomplexen wird bewußt verzichtet.

3. Die Hauptarbeitsstufen der Vorbereitung

3.1. Die Problemanalyse

Die erste Hauptarbeitsstufe wollen wir als Problemanalyse bezeichnen. Sie umfaßt die Arbeitsstufen
- Formulieren der Problemstellung,
- mathematische Modellierung,
- Ausarbeiten von Algorithmen.

Inhalt und Umfang dieser drei Arbeitsstufen werden im Abschnitt 4 erläutert.

3.2. Die Programmierung

In der zweiten Hauptarbeitsstufe Programmierung fassen wir die Arbeitsstufen
- Planung des Programmablaufes,
- Aufstellen der Programme,
- Testen der Programme an der EDV-Anlage

zusammen.
Auf diese Arbeitsstufen werden wir in den Lektionen 16 bis 20 eingehen.

3.3. Die Nutzrechnung

Zur dritten Hauptarbeitsstufe Nutzrechnung gehören die Arbeitsstufen
- Vorbereitung der Eingangsdaten,
- Lösung der Aufgaben mit Hilfe der EDV-Anlage,
- Auswertung der Ergebnisse.

Die Probleme der Nutzrechnung werden in der Lektion 21 behandelt.

Hauptarbeitsstufen	Problemanalyse	Programmierung	Nutzrechnung
Arbeitsstufe	Problemstellung	Programmablaufplanung	Vorbereitung der Ausgangsdaten
	Mathematisches Modell	Programmaufstellung	Lösung mit EDV-Anlagen
	Algorithmen	Programmtestung	Auswertung der Ergebnisse

Abbildung 1
Die Hauptarbeitsstufen der Vorbereitung von Aufgaben für die EDV

Die innerhalb der Hauptarbeitsstufen angegebenen Arbeitsstufen können mitunter einen geringen Arbeitsumfang aufweisen, wenn z. B. auf bereits ausgearbeitete Modelle oder Programme zurückgegriffen werden kann. Bei sehr einfachen Aufgabenstellungen können auch einige Stufen ausgelassen werden.

4. Die Arbeitsstufen der Problemanalyse

4.1. Die Formulierung der Problemstellung

Die erste Arbeitsstufe bei der Vorbereitung von Aufgaben, die mit Hilfe einer EDV-Anlage gelöst werden sollen, beginnt mit dem Herausarbeiten und Formulieren der Problemstellung. Da eine ausführliche Behandlung der Etappen der Einsatzvorbereitung, die vor der Programmierung zu durchlaufen sind, im II. Teil dieses Fernsehkurses vorgesehen ist, beschränken wir uns an dieser Stelle auf einige Grundsätze, soweit sie für die Darstellung des Zusammenhangs mit der Hauptarbeitsstufe Programmierung notwendig sind.
Bei Aufgabenstellungen geringen Umfanges kann die Problemstellung von einem Fachmann des jeweiligen Spezialgebietes erarbeitet und formuliert werden. Umfangreiche und komplexe Probleme werden durch größere Kollektive und in sozialistischer Gemeinschaftsarbeit vorbereitet.
Dabei ist mit größter Sorgfalt vorzugehen, da die Problemstellung den Ausgangspunkt für alle folgenden Arbeitsstufen darstellt. Jede nachträgliche Veränderung in der Problemstellung – und mag sie dem Außenstehenden noch so unwesentlich erscheinen – kann zu erheblichem Mehraufwand bei der Programmierung führen. Es ist deshalb unbedingt notwendig, bei der Erarbeitung der Problemstellung für eine Teilaufgabe vom komplexen Charakter der Aufgabe auszugehen und alle Beziehungen und Verknüpfungen zu anderen Teilaufgaben von vornherein zu berücksichtigen. Da die Einsatzvorbereitung im allgemeinen einen längeren Zeitraum umfaßt, muß bei der Erarbeitung der Problemstellung unbedingt von der perspektivischen Entwicklung des Teilkomplexes ausgegangen und die praktische Anwendung der marxistisch-leninistischen Organisationswissenschaft allseitig gesichert werden.
Die Problemstellung beinhaltet die textliche Darstellung der zu lösenden Aufgabe. Sie soll sich in der Form möglichst dem Schema für die Vorgabe von Mathematikaufgaben mit der Unterteilung nach gegebenen und gesuchten Größen anpassen. Wesentlicher Bestandteil der Problemstellung ist die eindeutige Fixierung des zu erzielenden Ergebnisses.

Die Leiter sind für die Einführung der EDV in ihrem Wirkungsbereich voll verantwortlich. Sie müssen deshalb bei der Herausarbeitung der Problemstellung richtungweisend mitwirken. Letztlich soll die elektronische Datenverarbeitung dazu beitragen, im Verantwortungsbereich des jeweiligen Leiters die Qualität der Planungs- und Leitungstätigkeit zu verbessern und die Effektivität der wissenschaftlichen Arbeit zu erhöhen.
Der Zeitaufwand für die Formulierung der Problemstellung hängt von dem zu untersuchenden Gegenstand ab. Bei umfangreichen Aufgaben ist es üblich, die Problemstellung in Form einer Studie zu beginnen.

4.2. Die mathematische Modellierung

> Unter einem Modell versteht man in der Wissenschaft die Abbildung komplizierter und komplexer Strukturen und Funktionen eines Gegenstandsbereichs (z. B. der Ökonomie) auf einfachere und übersichtlichere Strukturen des gleichen oder eines anderen Gegenstandsbereiches (z. B. der Mathematik).[1]

Unter der Arbeitsstufe "Mathematische Modellierung" wollen wir eine Abbildung des in der ersten Arbeitsstufe textlich formulierten Problems auf eine mathematische Struktur verstehen.
Die mathematische Struktur kann dabei verschiedenartig sein. Bei einfachen Aufgabenstellungen beinhaltet die mathematische Struktur lediglich eine formelmäßige Darstellung, aus der die gesuchte Lösung unmittelbar berechnet werden kann. Für Aufgaben aus dem Bereich der theoretischen Physik entsprechen der mathematischen Struktur häufig Systeme von Differentialgleichungen. Bei der mathematischen Modellierung von Planungs- und Leitungssystemen der sozialistischen Volkswirtschaft ergeben sich lineare Ungleichungssysteme, Netzplandarstellungen u. a.
Mit der Ausarbeitung des mathematischen Modells muß nicht in jedem Falle das Lösungsverfahren bekannt sein.

4.3. Das Aufstellen von Algorithmen

Die letzte Arbeitsstufe der Problemanalyse beinhaltet das Aufstellen von Algorithmen zur Bestimmung des Lösungsverfahrens des mathematischen Modells.

> Unter einem Algorithmus verstehen wir eine Vorschrift, die aus einer eindeutig bestimmten, endlichen Folge von Grundoperationen besteht und zur schrittweisen Lösung einer Klasse von Aufgaben dient.

Einfache Beispiele für Algorithmen sind die Regeln für die Durchführung der vier Grundrechenarten im Dezimalsystem, die Verfahren zur Lösung von linearen Gleichungssystemen oder zum Berechnen der Quadratwurzel. Charakteristisch für diese Algorithmen ist, daß komplizierte Aufgaben auf eine schrittweise Durchführung einfacher Operationen zurückgeführt werden. Die Addition einer Reihe mehrstelliger Zahlen wird bekanntlich auf die Addition

1 Fachwörterbuch – Begriffe und Sinnbilder der Datenverarbeitung, Schriftenreihe Datenverarbeitung, Verlag Die Wirtschaft, Berlin 1968.

einstelliger Zahlen zurückgeführt. Dieses Prinzip der Zerlegung von komplizierten Aufgaben in Grundoperationen haben wir bei der technischen Realisierung von Rechenoperationen mit Hilfe einer EDV-Anlage in den Themenkomplexen 2 und 3 kennengelernt.
Für das Aufstellen von Algorithmen ist Voraussetzung, daß die Grundoperationen, die zur Realisierung der Einzelschritte zur Verfügung stehen, definiert sind. Die Grundoperationen, die auf einer EDV-Anlage ausgeführt werden können, sind bereits bekannt. Wir stellen sie in der symbolischen Schreibweise nochmals zusammen:

$M = \{ +, -, x, /, <, > \}$

M = Menge der Grundoperationen
\+ Addition
\- Subtraktion
x Multiplikation
/ Division
< Kleinerbeziehung
> Größerbeziehung

Beim Aufstellen von Algorithmen für die folgenden Beispiele wollen wir die Menge M als Menge der Grundoperationen betrachten. Damit ist es möglich, Algorithmen, die auf diesen Grundoperationen aufbauen, von einer EDV-Anlage ausführen zu lassen.

5. Das Erarbeiten der Problemanalyse

5.1. Die Berechnung von Kurvenscheiben

In Erzeugnissen des Maschinenbaus, wie in Werkzeugmaschinen, Verpakkungsautomaten u. a., werden Kurvenscheiben zur Steuerung von Bewegungsvorgängen benutzt. Die Kontur der Kurvenscheibe kann mit Hilfe von Grundgesetzen der Geometrie und der Getriebelehre auf zeichnerischem oder rechnerischem Wege ermittelt werden.

Problemstellung

Für die Herstellung von Kurvenscheiben zur Steuerung des Bewegungsvorganges des Fräsers einer Werkzeugmaschine soll eine Tabelle aufgestellt werden, die für jeden Punkt des Frästisches die Koordinaten der Kurvenscheibe angibt. Abbildung 2 zeigt die Prinzipskizze der Problemstellung.
Die Punkte P_1 des Koordinatensystems des Frästisches werden in Polarkoordinaten durch den Radius r und den Winkel φ bestimmt.
Die Koordinaten der Punkte P_2 der Kurvenscheibe werden in rechtwinkligen Koordinaten x und y angegeben. Der Punkt G stellt einen Gelenkpunkt des vereinfacht dargestellten Übertragungsmechanismus dar.

Die Kurzfassung des Problems lautet:

Gegeben
sei in Form von technischen Zeichnungen der vollständige Übertragungsmechanismus zwischen den beiden Koordinatensystemen.

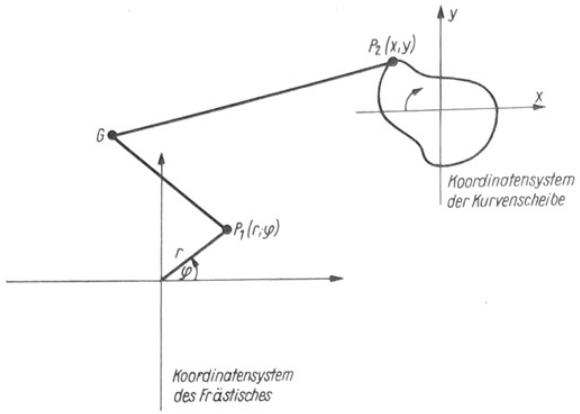


Abbildung 2
Prinzipskizze für Kurvenscheiben-berechnung

Gesucht

ist der mathematische Zusammenhang zwischen beiden Koordinatensystemen, wobei r und φ als gegebene und x und y als gesuchte Größen zu betrachten sind.

Mathematische Modellierung

Auf die Ableitung der mathematischen Darstellung muß hier verzichtet werden, da die Prinzipskizze dafür nicht ausreicht. Die Grundlage dafür bilden Konstruktionszeichnungen und Grundgesetze der Geometrie und Getriebelehre.
Wir gehen von dem Ergebnis aus:

$$x = \frac{90 - r\cos\varphi}{N} \qquad y = \frac{90 - \frac{r}{2}(2 \cdot \sin\varphi + \cos\varphi - r)^2}{N}$$

wobei

$$N = \left[(90 - r\cos\varphi)^2 + (1 - r\sin\varphi)^2\right]^2.$$

Das mathematische Modell wird durch eine formelmäßige Darstellung repräsentiert. Eine besondere Aufstellung eines Algorithmus zur numerischen Berechnung der gesuchten Werte ist nicht erforderlich, da aus der Formeldarstellung die Reihenfolge der Grundoperationen unmittelbar ersichtlich ist. Wir weisen aber darauf hin, daß in der Menge M der gegebenen Grundoperationen die trigonometrischen Funktionen sin φ und cos φ explizit nicht enthalten sind. Aus Lektion 14 wissen wir, daß bei der Anwendung einer Datenverarbeitungsanlage die Werte mit einem Algorithmus berechnet werden, der in einem Bibliotheksprogramm erfaßt ist und nur die Grundoperationen der Menge M umfaßt (Reihenentwicklung).

5.2. Die Lohnberechnung

Betrachten wir nun ein stark vereinfachtes Beispiel aus der Lohnabrechnung.

Problemstellung

Für Stundenlöhne sind der monatliche Bruttolohn und der SV-Beitrag zu berechnen.

Gegeben:

SL = Stundenlohn
A = Anzahl der geleisteten Stunden je Monat

Gesucht:

B = Bruttolohn
SV = Sozialversicherungsbeitrag

Mathematische Modellierung

Die mathematische Modellierung ist hier sehr einfach, sie besteht aus einigen Formeln.
Der Bruttolohn errechnet sich aus der Anzahl der im Monat geleisteten Stunden A multipliziert mit dem Stundenlohn SL.

$$B = A \cdot SL.$$

Für den SV-Beitrag gilt:
Wenn der Bruttoverdienst kleiner als 600 Mark ist, dann beträgt der SV-Beitrag 10 % des Bruttoverdienstes, anderenfalls 60 Mark.

Kürzer schreiben wir dafür:

Wenn $B < 600$, dann $SV = 0{,}1 \cdot B$,
anderenfalls $SV = 60$.

Auch bei diesem Beispiel ist die Problemanalyse hiermit abgeschlossen. Die angegebenen Formeln enthalten nur Grundoperationen und entsprechen somit den Forderungen des Algorithmus.

5.3. Die Umsatzberechnung

Problemstellung

Für ein Warenhaus mit n verschiedenen Waren ist täglich der Umsatz zu ermitteln und unter Berücksichtigung der Bestandskontrolle die Einhaltung der Mindestbestände zu kontrollieren. Wird der Mindestbestand unterschritten, ist eine Bestellung auszulösen.

Gegeben:

P_i = Preis der Ware i, wobei $i = 1, 2, \ldots, n$.
A_i = Anzahl der täglich verkauften Waren i.

Gesucht:

U = Umsatz je Tag.

Für die Kontrolle der Mindestbestände muß der Warenfluß zwischen Zulieferung, Lagerung und Verkauf analysiert werden. Eine schematische Darstellung des Warenflusses zeigt Abbildung 3.

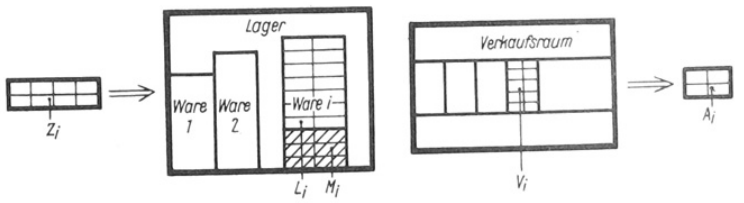


Abbildung 3
Schematische Darstellung des Warenflusses während der Verkaufszeit

Gegeben:

Z_i = Zugänge der Ware i während der Verkaufszeit
L_i = Bestand der Ware i im Lager vor Verkaufsbeginn
V_i = Bestand der Ware i im Verkaufsraum vor Verkaufsbeginn
A_i = verkaufte Anzahl der Ware i während der Verkaufszeit
M_i = geforderte Mindestbestände der Ware i

Gesucht:

B_i = Bestand der Ware i zum Zeitpunkt des Geschäftsschlusses.

Mathematische Modellierung

Wir beginnen mit der Ermittlung des Umsatzes; er ergibt sich aus der Summe der Produkte der Preise und der Anzahl der verkauften Waren. Es gilt:

$$U = P_1 \cdot A_1 + P_2 \cdot A_2 + \dots P_n \cdot A_n.$$

Mit Hilfe des Summenzeichens schreiben wir dafür

$$U = \sum_{i=1}^{n} A_i \cdot P_i.$$

Unter Beachtung von Abbildung 3 ergibt sich für den Bestand B_i zum Zeitpunkt des Geschäftsschlusses:

$$B_i = Z_i + L_i + V_i - A_i \qquad i = 1, 2, \dots, n.$$

Die rechtsstehende Angabe $i = 1, 2, \dots, n$ weist darauf hin, daß diese Berechnung für alle Waren i auszuführen ist.
Beachten wir nun die Einhaltung der Mindestbestände.
In Abbildung 3 ist ein Teil der im Lager befindlichen Anzahl L_i schraffiert dargestellt. Dieser Teil sei der Mindestbestand M_i, wird er unterschritten, ist eine neue Bestellung auszulösen.
Die mathematische Darstellung für diesen Sachverhalt lautet:

Wenn $B_i < M_i$, dann Bestellung für Ware i auslösen.

$i = 1, 2, \dots, n.$

Die algorithmische Darstellung fällt mit der mathematischen Modellierung zusammen, da die einzelnen Beziehungen nur Grundoperationen der Menge M enthalten.

5.4. Die Berechnung der Quadratwurzel

Häufig erfordert eine mathematische Aufgabe das Berechnen der Quadratwurzel. Der Mensch benutzt dazu entweder den Rechenstab oder, wenn eine höhere Genauigkeit gefordert wird, ein Tabellenbuch. In der Rechentechnik ist es nicht üblich, den kostbaren Speicherplatz mit den Werten des Tabellenbuches zu belegen. Da aber eine EDV-Anlage nur die vier Grundrechenarten ausführen kann, ergibt sich folgende

Problemstellung

Gesucht ist ein Verfahren zur Berechnung der Quadratwurzel, das nur die vier Grundrechenarten benutzt.

Gegeben

ist der Radikand $a > 0$.

Gesucht

ist der Wert für $x = \sqrt{a}$ unter Benutzung der Grundoperationen der Menge M.

Mathematische Modellierung

Schreibt man für $x = \sqrt{a}$, indem man diesen Ausdruck auf beiden Seiten quadriert, $x^2 = a$ und vergleicht damit die Funktionen $f(x) = x^2 - a$, so ist der gesuchte Wert x mit der positiven Nullstelle der Funktion $f(x) = x^2 - a = 0$ identisch, weil diese die Bedingung $x^2 = a$ erfüllt.
Die mathematische Darstellung lautet in dieser Form:

Ermittle die positive Nullstelle der Funktion

$$f(x) = x^2 - a.$$

Aufstellen des Algorithmus

Zur Bestimmung der Nullstellen von Funktionen sind in der Mathematik eine Reihe von Näherungsverfahren ausgearbeitet. Im Rahmen dieser Lektion ist es nicht möglich, näher auf diese Verfahren einzugehen. Wir wollen hier nur das Ergebnis darstellen.
Die Näherungsbeziehung für die Ermittlung der Nullstelle unseres Beispieles lautet

$$x_i = \frac{1}{2}\left(x_{i-1} + \frac{a}{x_{i-1}}\right) \qquad i = 1, 2, \dots$$

Bei dieser Formel handelt es sich um eine Näherungsbeziehung, die eine schrittweise Berechnung der Quadratwurzel gestattet. Ein solches Verfahren wird Iterationsverfahren genannt.
Die nach jedem Schritt ermittelten Werte wollen wir mit x_1, x_2, x_3... allgemein mit x_i bezeichnen. In der Mathematik bezeichnet man diese Größen als Glieder einer Zahlenfolge. In unserem Fall hat die Zahlenfolge die Eigenschaft, daß sie sich mit wachsendem immer mehr dem genauen Wert von $\sqrt{a}$ nähert.

Beispiel

Zu berechnen ist die Quadratwurzel aus a = 2.
Die Lösung erhalten wir durch das Aufstellen der Zahlenfolge. Die einzelnen Glieder ergeben sich aus der genannten Formel, wobei für a der Wert 2 einzusetzen ist, der Index i durchläuft der Reihe nach die Werte $1, 2, 3, \dots$.

Für i = 1 gilt

$$x_1 = \frac{1}{2}\left(x_0 + \frac{2}{x_0}\right)$$

x_0 ist der Anfangswert, der eine beliebige positive Zahl sein kann. Für die Anzahl der auszuführenden Schritte ist die Wahl des Anfangswertes mitbestimmend. Um mit wenig Schritten auszukommen, ist es von Vorteil, wenn der Anfangswert in der Nähe des Ergebnisses liegt.

1. Schritt: Wir setzen $x_0 = 1$ und erhalten

$$x_1 = \frac{1}{2}\left(1 + \frac{2}{1}\right) = \frac{1}{2} \cdot 3 = \frac{3}{2} = 1{,}5.$$

2. Schritt: Es gilt i = 2, und es folgt:

$$x_2 = \frac{1}{2}\left(x_1 + \frac{2}{x_1}\right)$$

$$x_2 = \frac{1}{2}\left(\frac{3}{2} + \frac{2 \cdot 2}{3}\right) = \frac{1}{2} \cdot \frac{17}{6} = \frac{17}{12} = 1,4166$$

3. Schritt: Für i = 3 ergibt sich:

$$x_3 = \frac{1}{2}\left(x_2 + \frac{2}{x_2}\right)$$

$$x_3 = \frac{1}{2}\left(\frac{17}{12} + \frac{2 \cdot 12}{17}\right) = \frac{1}{2} \cdot \frac{577}{204} = \frac{577}{408} = 1,4142$$

Wir könnten den für x_3 berechneten Wert wieder in die Formel einsetzen und die Berechnung fortsetzen. Um festzulegen, wann die Berechnung abgebrochen werden kann, ist eine zusätzliche Bedingung erforderlich. Sie kann aus den Genauigkeitsforderungen für die zu berechnende Wurzel abgeleitet werden.
Eine Schranke für die erreichte Genauigkeit erhält man in unserem Falle aus der Differenz zweier aufeinanderfolgender Glieder der Zahlenfolge.
Allgemein schreibt man eine solche Bedingung in der Form

$$x_i - x_{i-1} < \varepsilon$$

wobei ε eine Schranke für die Genauigkeit darstellt.
Im Beispiel wollen wir $\varepsilon = 0,005$ fordern.
Nach den einzelnen Schritten erhalten wir folgende Werte:

i = 1,
$|x_1 - x_0| = |1,5 - 1| = 0,5$

i = 2
$|x_2 - x_1| = |1,4166 - 1,5| = 0,0834$

i = 3
$|x_3 - x_2| = |1,4142 - 1,4166| = 0,0024$

Der Wert für i = 3 ist kleiner als 0,005 und bestimmt somit den Abbruch der Iteration.
Der erhaltene Näherungswert lautet $\sqrt{2} = 1,4142$.

Verfolgen wir die einzelnen Schritte der Wurzelberechnung, stellen wir fest, daß nur Grundoperationen der Menge M benutzt wurden. Dieses Verfahren kann somit von einer elektronischen Datenverarbeitungsanlage bearbeitet werden.

Anhand einiger zum besseren Verständnis sehr vereinfachter Beispiele haben wir uns mit der Erarbeitung der Problemanalyse beschäftigt. In der Praxis müssen für wesentlich kompliziertere Aufgaben Problemstellungen formuliert, mathematische Modelle und Algorithmen erarbeitet werden.

Die Hauptarbeitsstufe Problemanalyse steht am Beginn der Arbeiten zur Lösung von Verarbeitungsaufgaben mit Hilfe einer elektronischen Datenverarbeitungsanlage. Die Qualität der Arbeit dieses Stadiums entscheidet jedoch mit über die ökonomische Nutzung dieser Technik und beeinflußt weitgehend ihren Einsatz für die verschiedensten Aufgaben. Die Problemanalyse kann deshalb niemals von den Mitarbeitern der Organisations- und Rechenzentren allein erarbeitet werden. Sie muß in Gemeinschaftsarbeit unter Mitwirkung der Wirtschaftsfunktionäre und der Leiter der Bereiche, in denen die Datenverarbeitung eingeführt werden soll, entstehen. Nur so wird es gelingen, mit Hilfe elektronischer Datenverarbeitungsanlagen die Planungs- und Leitungstätigkeit in den einzelnen Verantwortungsbereichen zu verbessern und die Effektivität der wissenschaftlichen Arbeit zu erhöhen.

LEKTION 15

LEKTION

16

Die Programmablaufplanung (I)

1. Einleitung

2. Die Darstellung von Programmablaufplänen

2.1. Die Sinnbilder für Programmablaufpläne
2.1.1. Die Programmlinie
2.1.2. Die Ergibtanweisungen
2.1.3. Das Sinnbild für allgemeine Operationen
2.1.4. Das Sinnbild für die Ein- und Ausgabe von Informationen in EDV-Anlagen
2.1.5. Die Sinnbilder für den Programmanfang und das Programmende
2.2. Die Darstellung eines einfachen Programmablaufplans

3. Die Verwendung von Unterprogrammen

3.1. Das Sinnbild für Unterprogramme
3.2. Die Darstellung eines Programmablaufplans zur Berechnung von Führungskurven

4. Die Verzweigung eines Programmablaufes

4.1. Das Sinnbild für Testanweisungen und Vergleiche
4.2. Die Darstellung eines Programmablaufplans zur Berechnung des SV-Beitrages
4.3. Die Darstellung eines Programmablaufplans zur Ermittlung der kleinsten von drei Zahlen
4.3.1. Das Aufstellen des Programmablaufplans
4.3.2. Die Numerierung der einzelnen Programmschritte
4.3.3. Der Trockentest

5. Die Arten der Programme

1. Einleitung

LEKTION

16

Aus Lektion 15 ist bekannt, daß zu jeder Aufgabe, die man auf einer elektronischen Datenverarbeitungsanlage lösen will, ein Algorithmus erforderlich ist. Mit dem Aufstellen des Algorithmus sind die Vorbereitungen aber noch nicht abgeschlossen.
Die Datenverarbeitungsanlage arbeitet nach einem vom Menschen vorgegebenen Programm, das aus einzelnen Anweisungen, den Befehlen, besteht. Um für eine bestimmte Aufgabe ein Programm entwickeln zu können, muß man sich über den anzuwendenden Algorithmus und den logischen und zeitlichen Ablauf seiner einzelnen Teile Klarheit verschaffen. Zur übersichtlichen Darstellung von Algorithmen dienen Programmablaufpläne.
In dieser und in der nächsten Lektion befassen wir uns mit der Aufstellung von Programmablaufplänen. Wir werden die notwendigen Sinnbilder kennenlernen und an einfachen Beispielen erläutern.

2. Die Darstellung von Programmablaufplänen

Programmablaufpläne veranschaulichen den inneren, durch Operationen zu beschreibenden Ablauf der Organisation in informationsverarbeitenden Systemen.

> Der Programmablaufplan (PAP) ist eine problem- oder maschinenorientierte Beschreibung des Algorithmus, nach dessen allgemeinen oder detaillierten Vorschriften auf der EDV-Anlage aus den verfügbaren Eingabedaten die gewünschten und möglichen Ergebnisdaten erzeugt werden.[1]

Ein Programmablaufplan ist graphisch übersichtlich zu gestalten, der Aufwand zum Zeichnen möglichst gering zu halten.
Es gibt verschiedene Möglichkeiten für die zeichnerische Darstellung von Programmablaufplänen. Sie sind im DDR-Standard TGL 22 451 vom 23. Oktober 1967 festgelegt. Am weitesten verbreitet ist die Kästchenmethode, mit der wir uns daher eingehend beschäftigen wollen.

2.1. Die Sinnbilder für Programmablaufpläne

Programmablaufpläne sind im wesentlichen aus Sinnbildern für Operationen und Sinnbildern der Programmlinie zusammengesetzt.

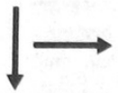
> Sinnbilder sind einheitliche graphische Figuren mit festgelegten Bedeutungen als Bausteine für Programmablaufpläne, mit deren Hilfe Organisationsabläufe und Programmabläufe erläutert und veranschaulicht werden.[1]

1 Fachwörterbuch – Begriffe und Sinnbilder der Datenverarbeitung, Schriftenreihe Datenverarbeitung, Verlag Die Wirtschaft, Berlin 1968.

LEKTION 16

2.1.1. Die Programmlinie

Ein wesentliches Element jedes Programmablaufplanes ist die Programmlinie, die den Ablauf des Programms angibt. Sie wird auch als Leitlinie bezeichnet. In der Regel verläuft sie von oben nach unten. Sie wird durch die einzelnen Sinnbilder, die die Merkmale der Operationen aufnehmen, unterbrochen.



Programmlinie

2.1.2. Die Ergibtanweisungen

Mit Hilfe des Ergibtzeichens werden Ergibtanweisungen gebildet. Das Ergibtzeichen wird aus Doppelpunkt und Gleichheitszeichen zusammengesetzt.

:= Ergibtzeichen

Beispiele für Ergibtanweisungen

1. x := a + 3
   Zu einer beliebigen bekannten Zahl a, wird 3 addiert. Das Ergebnis wird mit x bezeichnet. Hat a den Wert 5, so ergibt sich für x der Wert 8.
2. n := 1
   Der Größe n wird der Wert 1 zugeordnet.
3. i := i + 1
   Zum Wert i wird 1 addiert. Das Ergebnis ist der neue Wert i. Der Wert i wird also um 1 erhöht.
4. y := x (x - 5)
   Von der Größe x wird 5 subtrahiert und das Ergebnis mit x multipliziert. Das Endergebnis ist der neue Wert von y.
5. k := 2n + 3
   Der Wert n wird mit 2 multipliziert, zum Produkt wird 3 addiert und das Ergebnis k genannt.

■ Ergibtanweisungen sind keine Gleichungen.

Machen wir uns das an einem Beispiel klar:
a = b b = a
a := b b := a

Wenn wir die Gleichung a = b lesen, wissen wir, daß die Größen a und b denselben Wert haben.
Wir können dafür auch schreiben b = a.
Die beiden Seiten einer Gleichung sind also vertauschbar.

Die Ergibtanweisung a := b besagt: a bekommt den Wert von b, ganz gleich, wie groß a vorher ist.
Die umgekehrte Anweisung b := a bedeutet dagegen, daß b den Wert von a erhält, unabhängig davon, welchen Wert b vorher hat.
Die beiden Seiten einer Ergibtanweisung sind nicht vertauschbar.

Ganz deutlich wird der Unterschied zwischen einer Ergibtanweisung und einer Gleichung am Beispiel i := i + 1. Die Ergibtanweisung i := i + 1 besagt: Der Wert i wird um 1 erhöht.

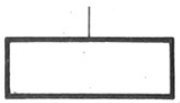
Eine Gleichung i = i + 1 hätte keine Berechtigung, denn keine Zahl stimmt mit einer um 1 größeren überein.

LEKTION 16

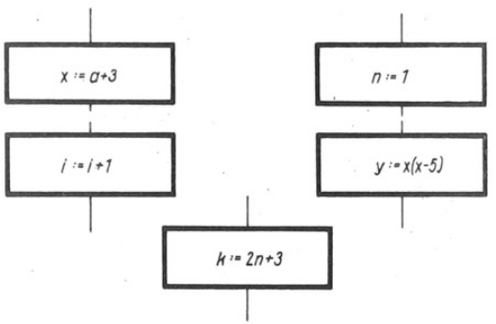
▌ Bei einer Ergibtanweisung wird aus den bekannten Größen auf der rechten Seite die links stehende Größe berechnet.

2.1.3. Das Sinnbild für allgemeine Operationen

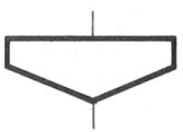
Das Sinnbild für allgemeine Operationen ist ein einfaches Rechteck, durch das die Programmlinie unterbrochen wird. In dieses Sinnbild werden die Ergibtanweisungen eingetragen.



Die angegebenen Ergibtanweisungen werden im Programmablaufplan in folgender Form geschrieben:

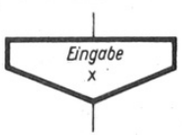


2.1.4. Das Sinnbild für die Ein- und Ausgabe von Informationen in EDV-Anlagen

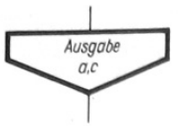


In das Sinnbild für Ein- und Ausgabeoperationen wird eingetragen, ob es sich um eine Eingabe oder eine Ausgabe handelt und welche Größen ein- oder ausgegeben werden.

Eingabe einer Größe x

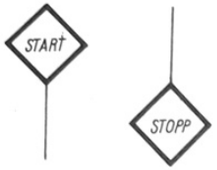


Ausgabe der Größen a und c



2.1.5. Die Sinnbilder für den Programmanfang und das Programmende

Für Anfang und Ende eines Programmablaufplans werden auf die Spitze gestellte Quadrate verwendet, in die START bzw. STOPP eingetragen wird. Bei dem Start-Symbol beginnt die Programmlinie, beim Stopp-Symbol endet sie.

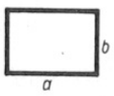


Handelt es sich um ein Unterprogramm, das der Programmbibliothek zu entnehmen ist, so werden für Beginn und Ende folgende Sinnbilder verwendet:

2.2. Die Darstellung eines einfachen Programmablaufplans

Aufgabe: Von einem Rechteck mit den Seitenlinien a und b sind Fläche und Umfang zu berechnen. Die Werte für die Seitenlinien a und b werden eingegeben.

Der Algorithmus lautet:



Fläche: $F = a \cdot b$
Umfang: $U = 2(a+b)$

Wir beginnen mit dem Start-Symbol, schreiben dann das Sinnbild für die Eingabe von a und b. Aus beiden Werten wird die Fläche und in der nächsten Ergibtanweisung der Umfang berechnet. In das folgende Sinnbild wird die Ausgabe eingetragen und mit dem Stopp-Symbol der Programmablaufplan beendet.

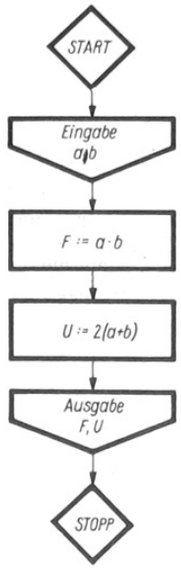


Abbildung 1
Programmablaufplan zur Berechnung von Fläche und Umfang eines Rechtecks

LEKTION 16

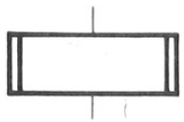
3. Die Verwendung von Unterprogrammen

Für jede Datenverarbeitungsanlage besteht eine umfangreiche Programmbibliothek, in der Unterprogramme für Standardaufgaben, wie die Berechnung einer Wurzel oder des Sinus oder Kosinus einer beliebigen Zahl, enthalten sind, die man in seinen eigenen Programmen verwenden kann. Auf diese Weise wird wertvolle Programmierzeit eingespart.

> Ein Unterprogramm ist ein Programm, das in sich selbst nicht vollständig ist und meist an mehr als einer Stelle in einem anderen Programm, dem Hauptprogramm, verwendet bzw. in andere Programme eingearbeitet werden kann.[2]

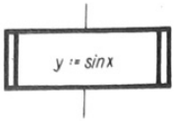
3.1. Das Sinnbild für Unterprogramme

Im Programmablaufplan drückt man die Verwendung solcher Standardprogramme durch das Sinnbild für Unterprogramme oder Programmteile aus.



Das Sinnbild ähnelt dem für allgemeine Operationen; allerdings sind die beiden senkrechten Begrenzungslinien doppelt ausgezogen. In das Sinnbild trägt man die entsprechende Ergibtanweisung ein, z. B. bedeutet:

2 Fachwörterbuch – Begriffe und Sinnbilder der Datenverarbeitung, Schriftenreihe Datenverarbeitung, Verlag Die Wirtschaft, Berlin 1968.



daß von einer Zahl x der Sinus zu ermitteln ist. Der Wert sin x wird mit y bezeichnet.

3.2. Die Darstellung eines Programmablaufplans zur Berechnung der Führungskurven

Stellen wir nun für die in Lektion 15 behandelte Aufgabe der Führungskurvenberechnung einen Programmablaufplan auf.
Wir gehen von den dort angegebenen Formeln aus:

$$x = \frac{90 - r \cos \varphi}{N^2}$$

$$y = \frac{90 - \frac{r}{2}(2 \sin \varphi + \cos \varphi - r)^2}{N^2}$$

wobei gilt

$$N = (90 - r \cos \varphi)^2 + (1 - r \sin \varphi)^2.$$

Wir erkennen, daß der Sinus bzw. Kosinus des Winkels φ mehrfach auftritt.

Mit geeigneten Verfahren der numerischen Mathematik kann man die Berechnung von sin φ und cos φ auf die vier Grundrechenarten zurückführen, so daß sie mit der Datenverarbeitungsanlage vorgenommen werden kann. Auf die Verfahren selbst wollen wir hier nicht eingehen, da bei der Programmierung für Standardaufgaben keine eigenen Befehle zu schreiben sind.

Da wir den Sinus und Kosinus des Winkels φ mehrfach benötigen, speichern wir die Werte für sin φ und cos φ, die wir über Unterprogramme erhalten, um die Berechnung für den gleichen φ-Wert nicht mehrmals ausführen zu müssen.

Das Aufstellen des Programmablaufplans

Wir beginnen mit dem Start-Symbol;

es folgt das Eingabe-Symbol für die Eingabe von r und φ; Unterprogrammsymbol für die Berechnung von sin φ, das wir mit s bezeichnen; Unterprogrammsymbol für die Berechnung von cos φ, das wir k nennen.
Auf die Berechnung des Nenners von x und y wollen wir verzichten und annehmen, daß N^2 bekannt ist.

Als nächstes tragen wir die Ergibtanweisung zur Berechnung des Zählers von x, den wir abkürzend Z nennen, ein. Wir schreiben Z := 90 - rk und nutzen damit aus, daß wir in der vorangegangenen Anweisung den Kosinus von φ berechnet und k genannt haben. Damit ergibt sich für die gesuchte Größe $x := Z/N^2$.

Den Wert von x könnten wir jetzt ausgeben. Wir berechnen aber noch y und geben dann die Werte x und y gemeinsam aus. Da wir den Nenner als bekannt voraussetzen wollen, brauchen wir nur den Zähler von y zu berechnen. Wir beginnen mit der Ergibtanweisung für die runde Klammer und schreiben dafür $Z := 2 \cdot s + k - r$, indem wir die Klammer mit Z bezeichnen und mit s und k wieder die bereits berechneten Werte für sind φ und cos φ verwenden.

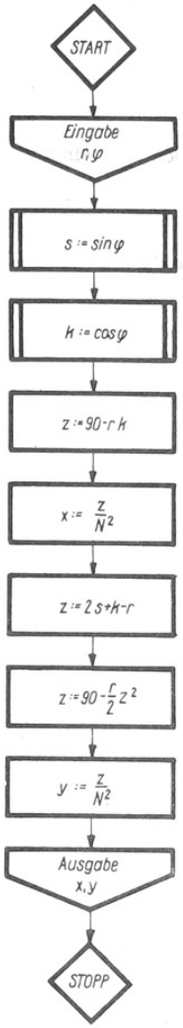


Abbildung 2
Programmablaufplan zur Berechnung der Führungskurven

Der Zähler ergibt sich dann durch $Z := 90 - \frac{r}{2} Z^2$. Damit können wir y mit der Anweisung $y := Z/N^2$ berechnen.

Die beiden Ergebniswerte x und y werden ausgegeben und der Programmablaufplan mit dem Stopp-Symbol abgeschlossen.

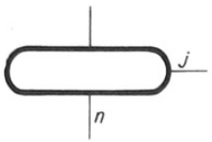
Das ist eine mögliche Lösung. In der Regel gibt es mehrere Varianten, um zum gleichen Ergebnis zu kommen. Wichtig ist, daß der Programmablaufplan den Verlauf der Rechnung möglichst übersichtlich und exakt angibt.

LEKTION 16

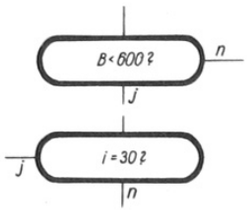
4. Die Verzweigung eines Programmablaufes

Aus den vorangegangenen Lektionen ist bekannt, daß eine elektronische Datenverarbeitungsanlage nicht nur umfangreiche Rechnungen mit hoher Geschwindigkeit ausführen, sondern auch Vergleiche vornehmen kann. Denken wir an die Berechnung des SV-Beitrages aus Lektion 15. Zur Lösung derartiger Aufgaben mit einer EDV-Anlage sind Vergleichsbefehle und bedingte Sprungbefehle erforderlich, deren Ergebnis eine Verzweigung des Programmablaufes ist.

4.1. Das Sinnbild für Testanweisungen und Vergleiche



Das Sinnbild für Vergleiche hat einen Eingang (oben) und zwei Ausgänge. Einer der beiden Ausgänge wird in der Regel nach unten gezeichnet, der andere wahlweise nach rechts oder links.



In das Sinnbild wird der Vergleich in Form einer Frage eingetragen, z. B.

Ist B kleiner als 600 ? oder
Ist i gleich 30 ?

Die Antwort auf eine solche Frage – man sagt auch auf einen solchen Test und spricht bei derartigen Ausdrücken von Testanweisungen – ist immer eindeutig: ja oder nein.
Die Programmlinie erfährt im Ergebnis eines solchen Tests eine Verzweigung. Die beiden Zweige werden mit j (ja) und n (nein) gekennzeichnet. Es ist freigestellt, welchen Ausgang man mit j und welchen man mit n bezeichnet.

4.2. Die Darstellung eines Programmablaufplans zur Berechnung des SV-Beitrages

Wir gehen von dem in Lektion 15 gefundenen Algorithmus aus:

$$B = A \cdot SL$$

$$SV = \begin{cases} 0{,}1 \cdot B, & \text{wenn } B < 600{,}\text{- M} \\ 60{,}\text{- M}, & \text{wenn } B \geq 600{,}\text{- M.} \end{cases}$$

Wir beginnen mit dem Start-Symbol und schreiben danach das Sinnbild für die Eingabe von Stundenlohn und Anzahl der Stunden im Monat.

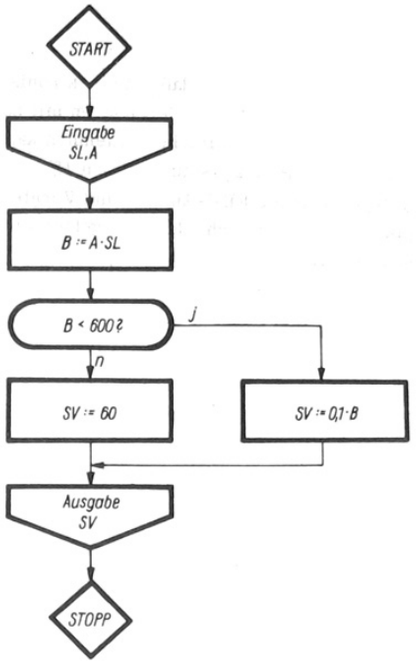


Abbildung 3
Programmablaufplan für die Ermittlung des SV-Beitrages

LEKTION 16

Zur Berechnung des Bruttolohnes B schreiben wir die Ergibtanweisung B := A · SL.
Es folgt eine Testanweisung, um zu ermitteln, ob der Bruttolohn unter 600,- M liegt oder nicht. Wir tragen also in das Sinnbild die Fragestellung ein B < 600 ? Es ergeben sich zwei Ausgänge, einer für die Antwort ja, den wir nach rechts wählen, und einer für nein, den wir nach unten zeichnen.

Verfolgen wir zunächst den senkrechten Zweig weiter.

Da die Frage B < 600 ? mit Nein beantwortet wurde, wissen wir, daß der SV-Beitrag 60,- M beträgt. Wir können also die Ergibtanweisung SV := 60 schreiben. Den so bestimmten SV-Beitrag geben wir aus. Danach schließen wir den Programmablaufplan mit dem Stopp-Symbol.

Verfolgen wir nun den Ja-Zweig.

In diesem Falle beträgt der SV-Beitrag 10 % des Bruttoverdienstes und ist nach der Formel SV := 0,1 · B zu berechnen, die wir als Ergibtanweisung schreiben. Der so ermittelte SV-Beitrag ist auszugeben und der Programmablaufplan mit dem Stopp-Symbol zu beenden. Beide Sinnbilder sind im Programmablaufplan schon enthalten. Wir nutzen das und führen die Programmlinien vor dem Ausgabe-Symbol wieder zusammen.

Es bleibt jedem selbst überlassen, anhand einiger Zahlenbeispiele die Richtigkeit des Programmablaufplans zu überprüfen. Nach der Verzweigung wird entweder die Anweisung SV := 60 oder die Anweisung SV := 0,1 · B ausgeführt, je nachdem, wie der Test ausgegangen ist.
Die Beispiele sollten so gewählt werden, daß möglichst alle Zweige des Programmablaufplans einmal durchlaufen und damit überprüft werden. Man spricht bei diesem Vorgehen von einem "Trockentest".

4.3. Darstellung eines Programmablaufplans zur Ermittlung der kleinsten von drei Zahlen

Aufgabe: Von drei gegebenen Zahlen a, b, c soll die kleinste gesucht werden.
Zur Aufstellung des Programmablaufplans müssen wir zunächst einen Algorithmus finden. Im Unterschied zu den behandelten Beispielen sind hier keine Rechenoperationen, sondern nur Vergleiche zwischen den vorgegebenen Zahlen a, b und c auszuführen.
Die gesuchte kleinste Zahl wollen wir mit MIN (Abkürzung für Minimum) bezeichnen. Wir erhalten das gesuchte Minimum durch Vergleich von jeweils zwei der Zahlen a, b, c.

4.3.1. Das Aufstellen des Programmablaufplans

Auf das Start-Symbol folgt das Sinnbild für die Eingabe von a, b und c.

Als erstes vergleichen wir a mit b, indem wir die Testanweisung a < b? schreiben. Den Nein-Ausgang wollen wir nach rechts, den Ja-Ausgang nach unten zeichnen.
Verfolgen wir zunächst den Ja-Ausgang.
Wir wissen jetzt: a ist kleiner als b, also kommt b für das gesuchte MIN nicht in Frage. Wir vergleichen nun a mit c und schreiben dafür a < c?

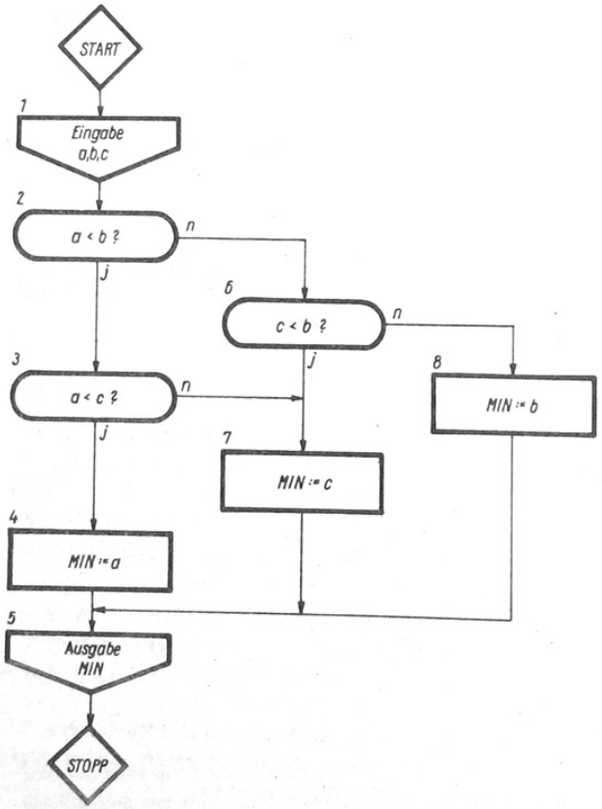


Abbildung 4
Programmablaufplan für die Ermittlung der kleinsten von drei Zahlen

Den Nein-Ausgang nehmen wir wieder nach rechts, den Ja-Ausgang nach unten.
Wir verfolgen zuerst wieder den Ja-Ausgang.
Wir wissen jetzt: a ist die kleinste Zahl, denn es war a < b und a < c. Wir können also die Ergibtanweisung MIN := a schreiben.
Danach folgt das Sinnbild für die Ausgabe von MIN und das Stopp-Symbol zum Abschluß des Programmablaufplans.
Verfolgen wir nun den Nein-Ausgang der letzten Testanweisung a < c?.
Wir wissen dann, daß c die kleinste der drei Zahlen ist.
Wir können also die Anweisung MIN := c schreiben und dann für die Ausgabe und das Programmende die schon vorhandenen Sinnbilder benutzen und die Programmlinien wieder zusammenführen.
Verfolgen wir nun den Nein-Ausgang der ersten Testanweisung a < b?.
Wir wissen, daß a als kleinste Zahl nicht in Frage kommt und müssen jetzt b mit c vergleichen. Dafür schreiben wir die Testanweisung c < b?. Als Ergebnis bekommen wir wieder eine Verzweigung. Bei "ja" wissen wir, daß c die kleinste Zahl ist, was wir durch die Ergibtanweisung MIN := c ausdrücken. Da wir diese Anweisung schon einmal geschrieben haben, werden wir den Ja-Ausgang der Testanweisung nach unten nehmen und mit der Programmlinie von vorhin zusammenführen.
Bleibt noch der Nein-Ausgang des Vergleiches c < b?. Da sowohl a < b? als auch c < b? mit Nein beantwortet wurden, ist b die kleinste Zahl, und wir schreiben die Ergibtanweisung MIN := b.
Für die Ausgabe von MIN und das Ende des Programmablaufplans benutzen wir die schon vorhandenen Sinnbilder, indem wir die Programmlinie nach dort führen.
Wir sehen an diesem Beispiel, daß in vielen Fällen ein Programmablaufplan gut geeignet ist, ohne Formeln einen Algorithmus eindeutig und übersichtlich darzustellen.

4.3.2. Die Numerierung der einzelnen Programmschritte

Oft ist zu einem Programmablaufplan eine Beschreibung anzufertigen, wobei die einzelnen Ablaufschritte erläutert werden. Um eine eindeutige Beziehung der Beschreibung zum Programmablaufplan herzustellen, ist es üblich, die einzelnen Sinnbilder des Programmablaufplans fortlaufend zu numerieren.
Die Zahlen werden an die linke obere Ecke der Sinnbilder geschrieben. Man beginnt mit Nummer 1 bei dem ersten Sinnbild nach dem Start-Symbol. Bei Testanweisungen wird in der Regel die Numerierung nach unten fortgesetzt. Ist man am Ende des Programmablaufplans angelangt, werden die "Nebenäste" numeriert.

4.3.2. Der Trockentest

Überzeugen wir uns anhand einiger Zahlenbeispiele von der Richtigkeit des aufgebauten Programmablaufplans zur Ermittlung der kleinsten von drei Zahlen, indem wir einen Trockentest ausführen.

Beispiele

1. a = 3, b = 9, c = 4
   Es werden die Ablaufschritte 1, 2, 3, 4, 5 durchlaufen.
   Ergebnis: MIN = 3
2. a = 8, b = 1, c = 13
   Es werden die Ablaufschritte 1, 2, 6, 8, 5 durchlaufen.
   Ergebnis: MIN = 1
3. a = 5, b = 0, c = -4
   Es werden die Ablaufschritte 1, 2, 6, 7, 5 durchlaufen.
   Ergebnis: MIN = -4
4. a = 1, b = 6, c = -5
   Es werden die Ablaufschritte 1, 2, 3, 7, 5 durchlaufen.
   Ergebnis: MIN = -5
5. a = 8, b = 8, c = 8
   Es werden die Ablaufschritte 1, 2, 6, 8, 5 durchlaufen.
   Ergebnis: MIN = 8

Der Trockentest bestätigt, daß alle Beispiele das richtige Ergebnis gebracht haben, wobei alle Zweige des Programmablaufplans mindestens einmal durchlaufen wurden.

5. Die Arten der Programme

Zum Schluß wollen wir noch auf ein wesentliches Merkmal der in dieser Lektion behandelten Programmablaufpläne hinweisen. Bei allen Beispielen handelt es sich um Geradeausprogramme.
Beim ersten Beispiel zur Berechnung von Fläche und Umfang eines Rechtecks führte die Programmlinie ohne Verzweigung vom Start- zum Stopp-Symbol, ebenso beim zweiten Beispiel der Führungskurvenberechnung.

Bei unverzweigten Geradeausprogrammen wird jede Anweisung genau einmal durchlaufen.

Bei den Beispielen der SV-Berechnung und der Ermittlung der kleinsten von drei Zahlen handelt es sich um verzweigte Geradeausprogramme.
Im Programmablaufplan für die SV-Berechnung werden alle Anweisungen einmal durchlaufen bis auf eine der beiden Anweisungen zur Festlegung des SV-Beitrages. Je nach der Höhe des Bruttolohnes wird entweder die Anweisung SV := 60 oder die Anweisung SV := 0,1 · B nicht durchlaufen.

Bei verzweigten Geradeausprogrammen wird jede Anweisung höchstens einmal durchlaufen.
Werden Anweisungen mehr als einmal durchlaufen, spricht man von zyklischen Programmen.

Beispiele zyklischer Programme werden wir in der folgenden Lektion behandeln.

Die Programmablaufplanung (II)

1. Einleitung
2. Die zyklischen Programme
2.1. Der Programmablaufplan für das Telefonieren
2.2. Der Programmablaufplan für die Berechnung des Minimums von n Zahlen
2.2.1. Das Bestimmen des Algorithmus
2.2.2. Das Aufstellen des Programmablaufplans
2.2.3. Der Trockentest
2.3. Der Programmablaufplan für die Umsatzberechnung eines Warenhauses
2.3.1. Das Bestimmen des Algorithmus
2.3.2. Das Aufstellen des Programmablaufplans
2.4. Der Programmablaufplan zur Berechnung einer Quadratwurzel
2.4.1. Das Bestimmen des Algorithmus
2.4.2. Das Aufstellen des Programmablaufplans
3. Abschließende Bemerkungen zur Aufstellung von Programmablaufplänen
3.1. Die Konnektoren
3.2. Methodische Hinweise
3.3. Die Leitlinienmethode
4. Die Zusammenstellung der wesentlichen Sinnbilder für Programmablaufpläne (Kästchenmethode)

1. Einleitung

In Lektion 16 lernten wir wesentliche Elemente und Sinnbilder zur Aufstellung von Programmablaufplänen kennen und wendeten sie an einfachen Geradeausprogrammen an.
Im Mittelpunkt dieser Lektion stehen Ablaufpläne für zyklische Programme. Im Unterschied zu Geradeausprogrammen werden bei zyklischen

LEKTION 17

Programmen bestimmte Anweisungen mehrfach durchlaufen. Wir werden dafür mehrere Beispiele durcharbeiten.
Am Schluß der Lektion stellen wir alle behandelten Sinnbilder für das Aufstellen von Programmablaufplänen zur besseren Übersicht nochmals zusammen.

2. Die zyklischen Programme

2.1. Der Programmablaufplan für das Telefonieren

Für eine nicht mathematische Aufgabe – den Vorgang des Telefonierens aus einer Telefonzelle – wollen wir einen Programmablaufplan aufstellen. Das ist möglich, wenn ein "Algorithmus" – eine Vorschrift aller dabei auszuführenden Handlungen unter Berücksichtigung der vorhandenen Möglichkeiten – angegeben werden kann.
Abbildung 1 zeigt eine mögliche Lösung dieser Aufgabe.
Der Programmablaufplan beginnt mit dem Start-Symbol.
Sinnbild 1 entspricht dem Suchvorgang im Fernsprechbuch.
Im Sinnbild 2 ist die Frage zu beantworten, ob die Fernsprechnummer gefunden wurde oder nicht.
Steht die Nummer nicht im Fernsprechbuch, müssen wir uns mit der Fernsprechauskunft in Verbindung setzen. Wir suchen zunächst die Fernsprechnummer der Auskunft, Sinnbild 3.
Haben wir die Nummer gefunden (Ja-Ausgang des Tests 2), brauchen wir Anweisung 3 nicht zu durchlaufen, sondern können zu den Ablaufschritten 4 und 5 übergehen: Handapparat abnehmen und Amtszeichen abhören. Ertönt kein Amtszeichen, ist das Telefon nicht benutzbar, und wir müssen den Handapparat auflegen, Sinnbild 13.
Ist das Amtszeichen vorhanden, so wird der Ablauf über die Anweisungen 6, 7 und 8 fortgesetzt. Nach dem Wählen der Fernsprechnummer warten wir auf das Freizeichen. Ertönt kein Freizeichen, so müssen wir den Handapparat auflegen (Sinnbild 14) und den Vorgang ab Anweisung 4 wiederholen.
Ertönt das Freizeichen, so müssen wir warten, ob sich der Teilnehmer meldet. Meldet er sich nicht, legen wir entsprechend Anweisung 13 den Handapparat auf.
Kommt eine Verbindung zustande, ergibt sich die Frage, ob sich der richtige Teilnehmer meldet, (Fragekästchen 10). Sind wir falsch verbunden, müssen wir über Anweisung 14 den Vorgang wiederholen.
Meldet sich der Teilnehmer, so müssen wir noch erfragen, ob es die Fernsprechauskunft ist oder der gewünschte Fernsprechteilnehmer, da davon unsere weiteren Handlungen abhängig sind. Im ersten Fall durchlaufen wir den linken Zweig, die Sinnbilder 16 und 15, und erfahren die gesuchte Fernsprechnummer von der Auskunft. Ist keine Fernsprechnummer angegeben, d. h. es besteht kein Fernsprechanschluß, müssen wir den Handapparat auflegen. Im anderen Fall ist das Programm über Anweisung 14 erneut zu durchlaufen.
Wird Test 11 mit Nein beantwortet, haben wir den gewünschten Teilnehmer erreicht und können sprechen (Sinnbild 12).

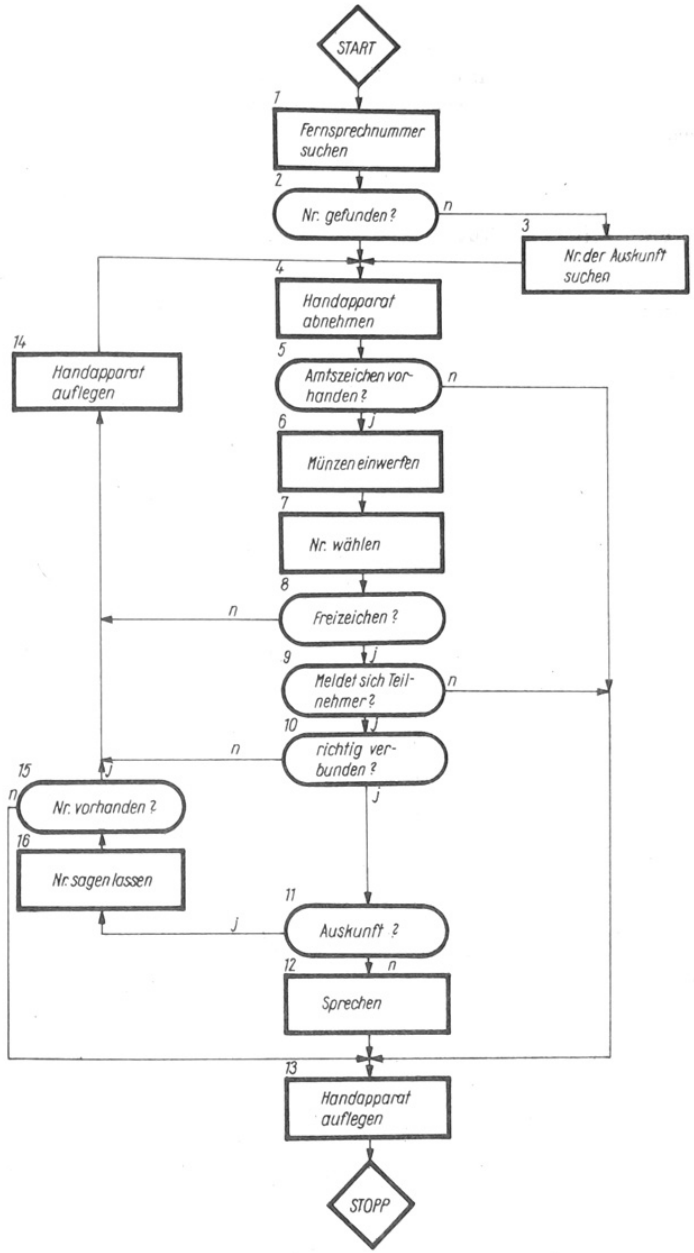


Abbildung 1
Programmablaufplan für das Telefonieren

Beim Trockentest ist darauf zu achten, daß alle Zweige durchlaufen und hinsichtlich ihrer Wirkungen untersucht werden.
Es gibt im vorliegenden Programmablaufplan Anweisungen, die mehrfach durchlaufen werden. Ertönt zum Beispiel in Anweisung 8 kein Freizeichen, gehen wir über das Kästchen 14 zum Kästchen 4 zurück und wiederholen die Anweisungen 4 bis 8. Man spricht hierbei von einem Zyklus. Der in

Abbildung 1 gezeigte Programmablaufplan weist noch mehrere Zyklen auf, die jeder selbst herausfinden wird.

Bei einem zyklischen Programm werden einzelne Anweisungen mehrfach durchlaufen.

Der Programmablaufplan in Abbildung 1 ist nicht vollständig. Um den Programmablaufplan übersichtlich zu gestalten, wurde nur eine Grobdarstellung gegeben. Es ist zum Beispiel nicht enthalten, daß man nach dem Auflegen des Handapparats in Anweisung 14 die Münzen wieder entnehmen kann. Es bleibt jedem überlassen, den Programmablaufplan nach Belieben zu vervollständigen.

2.2. Der Programmablaufplan für die Berechnung des Minimums von n Zahlen

Aufgabe: Aus einer Anzahl Zylinder mit gleichem Durchmesser aber unterschiedlicher Höhe ist der kürzeste herauszufinden.

2.2.1. Das Bestimmen des Algorithmus

Wir greifen wahllos einen Zylinder heraus und nehmen an, er sei der kürzeste. Um das nachzuweisen, müssen wir zeigen, daß alle anderen länger sind.
Wir nehmen einen zweiten Zylinder und vergleichen ihn mit dem ersten. Ist der zweite länger, bleibt der erste der kürzere. Danach wird ein dritter Zylinder zum Vergleich herangezogen.
Ist der zweite Zylinder kürzer als der erste, dann wird der erste beiseite gelegt und mit dem zweiten der dritte Zylinder verglichen.
Haben wir auf diese Weise alle Zylinder verglichen, haben wir den kürzesten ermittelt.

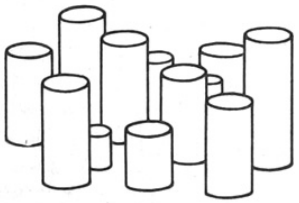


Abbildung 2
Zylinder mit gleichem Durchmesser aber unterschiedlicher Höhe

Diesen schrittweisen Vergleich wollen wir dem Programmablaufplan zur Ermittlung der kleinsten von n Zahlen $a_1, a_2, \ldots, a_n$ zugrunde legen.

2.2.2. Das Aufstellen des Programmablaufplans

Wir beginnen mit dem Start-Symbol und schreiben dann das Sinnbild für die Eingabe der n Zahlen a_1 bis a_n. Mit der nächsten Anweisung setzen wir a_1 als MIN, indem wir willkürlich die erste Zahl als kleinste annehmen.
Danach vergleichen wir die Zahlen a_2 bis a_n mit dieser "kleinsten" Zahl MIN.
Hierzu bauen wir folgenden Zyklus auf:
Wir bringen zunächst eine Zählgröße, die wir i nennen wollen, auf den

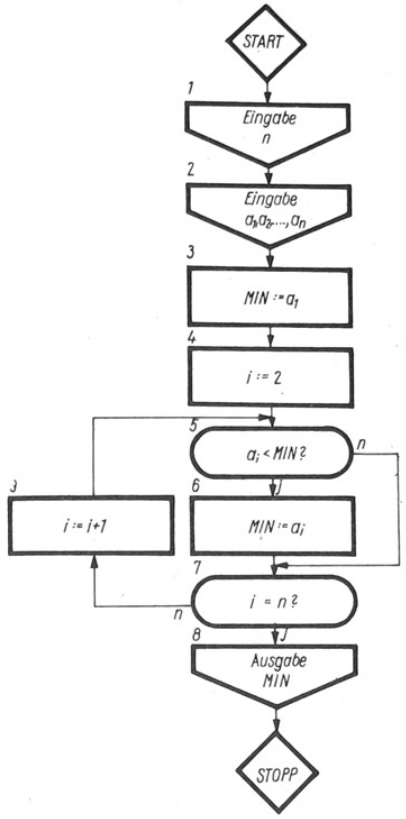


Abbildung 3
Programmablaufplan zur Ermittlung der kleinsten von n Zahlen

Wert 2, denn a_2 ist die erste Zahl, die wir mit MIN vergleichen wollen.
Die Testanweisung lautet a_i < MIN? Ist dies der Fall, ersetzen wir MIN durch a_i, denn a_i ist dann die kleinere Zahl. Ist a_i nicht kleiner als MIN, übergehen wir diese Anweisung.
Der weitere Ablauf des Algorithmus ist davon abhängig, ob die verglichene Zahl a_i schon die letzte Zahl a_n war. Im Programmablaufplan können wir das mit der Testanweisung i = n? erreichen. Ist i = n noch nicht erreicht, vergleichen wir die nächste Zahl mit MIN. Zu diesem Zweck erhöhen wir unsere Zählgröße i um 1. Wir führen den Nein-Zweig nach links heraus, schreiben die Ergibtanweisung i := i + 1 und treten vor der Testanweisung a_i < MIN? wieder in den Ablauf ein. Damit ist der Zyklus geschlossen.

Danach wird die nächste Zahl a_i mit MIN verglichen, denn i ist ja inzwischen um 1 größer. Ist diese Zahl a_i kleiner als MIN, wird sie als neues MIN verwendet, sonst behält MIN seinen alten Wert.
Es wird wieder gefragt: i = n? Ist dies nicht der Fall, wird i wieder um 1 erhöht, und der Zyklus beginnt von vorn.
Gilt schließlich i = n, haben wir mit MIN die kleinste Zahl gefunden und können diese ausgeben.
Der Programmablaufplan wird mit dem Stopp-Symbol abgeschlossen, und die einzelnen Anweisungen werden fortlaufend numeriert.

2.2.3. Der Trockentest

Verfolgen wir den Programmablaufplan an einem Zahlenbeispiel. Wir wählen

$a_1 = 5$
$a_2 = 8$
$a_3 = 3$
$a_4 = 4$
$a_5 = 2$
$a_6 = 9$

Es ist also hier $n = 6$.
Jeder erkennt sofort, daß 2 die kleinste Zahl ist.
Überprüfen wir nun den Programmablaufplan, indem wir einen Trockentest durchführen. Die Größen i und MIN wollen wir genau verfolgen und deshalb ihre Veränderungen aufschreiben.

i	MIN
2	5
3	3
4	2
5	
6	

Nach der Eingabe der sechs Zahlen bekommt in Anweisung 2 MIN den Wert 5, den wir in die Tabelle eintragen.
In Anweisung 3 bekommt i den Wert 3.
Am Ende hat i den Wert 6 und MIN den Wert 2. Der Trockentest bestätigt also die Richtigkeit des aufgestellten Programms.
Der Programmablaufplan ist für jeden Wert $n \geq 2$ richtig.

Die Anweisungen 4 bis 6 werden (n - 1)-mal durchlaufen. Es handelt sich um einen Zyklus mit fester Durchlaufzahl. Sobald n feststeht, ist auch bekannt, wie oft der Zyklus durchlaufen wird.

2.3. Der Programmablaufplan für die Umsatzberechnung eines Warenhauses

2.3.1. Das Bestimmen des Algorithmus

In Lektion 15 hatten wir einen Algorithmus für die Umsatzberechnung und Bestandskontrolle in einem Warenhaus gefunden. Für den ersten Teil der Aufgabe, die Umsatzberechnung, stellen wir den Programmablaufplan auf. Wir benutzen dazu die Formel

$$U = \sum_{i=1}^{n} A_i \cdot P_i$$

Die numerische Berechnung dieser Produktsumme erfordert eine Zerlegung in

$U_1 = A_1 \cdot P_1$
$U_2 = U_1 + A_2 \cdot P_2$
$U_3 = U_2 + A_3 \cdot P_3$
.
.
.
$U_n = U_{n-1} + A_n \cdot P_n$

Mit Ausnahme der ersten Zeile läßt sich folgende allgemeine Beziehung angeben:

$U_i = U_{i-1} + A_i \cdot P_i$
$i = 2, 3, \ldots, n$

Setzt man $U_0 = 0$, so gilt diese Beziehung auch für $i = 1$. Das zuletzt berechnete U_n wäre dann der gesuchte Umsatz U. Da in diesem Fall die Teilsummen $U_1, U_2, \ldots, U_{n-1}$ nicht interessieren, weil nur der Gesamtumsatz U berechnet werden soll, können wir die angegebene Formel wie folgt verändern:

$U = U + A_i P_i$
für $i = 1, 2, \ldots, n$
ausgehend von $U = 0$.

Damit können wir die Berechnung von U mit Hilfe eines Programm-Zyklus realisieren.

2.3.2. Das Aufstellen des Programmablaufplans

Wir gehen davon aus, daß die Preise P_i der n Artikel und die zugehörigen Stückzahlen A_i zu Beginn geschlossen in die EDV-Anlage eingegeben werden.

Wir zeichnen also nach dem Start-Symbol die zwei Sinnbilder für die Eingabe der Preise und der Stückzahlen.
Zur Vorbereitung des zyklischen Programmteils müssen wir die Ergibtanweisung U := 0 schreiben und dann, zur Festlegung des Ausgangspunktes von i, die Anweisung i := 1.
Nun können wir in den eigentlichen Zyklus zur schrittweisen Berechnung von U eintreten.
Zum Wert von U wird das Produkt $A_1 \cdot P_1$ addiert, denn i ist durch die Anweisung i := 1 gleich 1. Das Ergebnis – also das Produkt $A_1 \cdot P_1$ – ist der neue Wert von U. Nun ist die Berechnungsformel für $i = 1, 2, \ldots, n$ anzuwenden, das heißt, wir müssen eine Testanweisung anschließen, um zu erfragen, ob i schon den Wert n erreicht hat. Wir schreiben: Ist i = n?
Den Ja-Ausgang zeichnen wir nach unten, den Nein-Ausgang nach links.

Verfolgen wir zunächst den Nein-Ausgang.
Sicherlich wird am Anfang dieser Ausgang gewählt, denn i ist ja immer noch gleich 1. Wir müssen jetzt i um 1 erhöhen und die Berechnung nach der Formel wiederholen, womit wir den gewünschten Zyklus erhalten. Wir zeichnen die Programmlinie nach oben und schreiben die Anweisung $i := i + 1$, mit der wir das i auf den Wert 2 bringen. Die Programmlinien werden vor der Ergibtanweisung $U := U + A_i P_i$ wieder zusammengeführt.
Die Formel $U := U + A_i \cdot P_i$ besagt, daß zu U – dem Produkt $A_1 \cdot P_1$ – das Produkt $A_2 \cdot P_2$, denn es gilt inzwischen $i = 2$, addiert wird. Das Er-

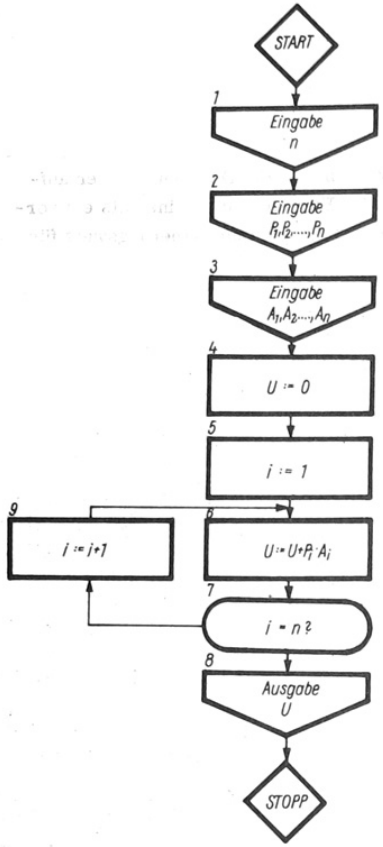


Abbildung 4
Programmablaufplan für die Umsatzberechnung eines Warenhauses

gebnis, also die Produktsumme $A_1P_1 + A_2P_2$, ist der neue Wert U. Es folgt wieder die Frage, i = n? Ist das nicht der Fall, wird der Ablauf beim Nein-Ausgang der Testanweisung fortgesetzt. i wird erneut um 1 erhöht, kommt also auf den Wert 3, und die Berechnungsformel wieder angewendet. Das wiederholt sich, bis i den Wert von n erreicht hat, der Test i = n? also mit Ja beantwortet wird. In diesem Falle entspricht die Größe U dem gesuchten Umsatz.
Wir können den Wert von U ausgeben und den Programmablaufplan mit dem Stopp-Symbol abschließen.
Wir haben es wieder mit einem Zyklus mit fester Durchlaufzahl zu tun. Ist n bekannt, weiß man, wie oft der Zyklus durchlaufen wird.

2.4. Der Programmablaufplan zur Berechnung einer Quadratwurzel

2.4.1. Das Bestimmen des Algorithmus

In Lektion 15 hatten wir einen Algorithmus für die Berechnung der Quadratwurzel aus einer beliebigen Zahl a gefunden. Zur Lösung auf der EDV-Anlage muß die Wurzelberechnung auf die vier Grundrechenarten zurückgeführt werden.

Das gelingt mit der Iterationsformel:

$$x_i = \frac{1}{2}\left(x_{i-1} + \frac{a}{x_{i-1}}\right)$$
$$i = 1, 2, \ldots$$
$$x_o = 1$$

Die Formel muß solange angewendet werden, bis die Differenz zweier aufeinander folgender Werte x_n und x_{n-1} dem Betrage nach kleiner als ein vorgegebenes ε ist. Das zuletzt berechnete x_n ist dann ein Näherungswert für die gesuchte $\sqrt{a}$.
Da nur das zuletzt berechnete x_n interessiert, nicht aber die zwischendurch erhaltenen x_i, können wir die Berechnungsvorschrift ein wenig abändern: Beginnend mit $x_o = 1$ berechnen wir

$$x_1 = \frac{1}{2}\left(x_o + \frac{a}{x_o}\right)$$

Gilt $|x_1 - x_o| \geq \varepsilon$, müssen wir die Berechnung fortführen. Hierzu setzen wir $x_o := x_1$ und berechnen nach der Formel ein neues x_1.
Gilt $|x_1 - x_o| < \varepsilon$, so ist das zuletzt berechnete x_1 ein Näherungswert für die gesuchte $\sqrt{x}$.

2.4.2. Das Aufstellen des Programmablaufplans

Wir eröffnen den Programmablaufplan mit dem Start-Symbol und schreiben dann das Sinnbild für die Eingabe von a.
Da wir aus negativen Zahlen keine Wurzel ziehen können, müssen wir in einer Testanweisung ermitteln, ob die eingegebene Zahl negativ ist. Den Ja-Ausgang zeichnen wir nach rechts.
Wird der Test a < 0? mit Nein beantwortet, können wir die Wurzel aus a ziehen. Wenn a = 0 oder a = 1 gilt, brauchen wir den Algorithmus nicht zu verwenden, da dann die Wurzelwerte ebenfalls 0 bzw. 1 sind. Wir schreiben also die Testanweisungen

a = 0?
und a = 1?

und nehmen beide Ja-Ausgänge nach rechts. In beiden Fällen können wir sofort x (die gesuchte Wurzel) gleich a setzen. Werden beide Tests mit Nein beantwortet, können wir den Algorithmus anwenden.
Als Ausgangspositionen für den Zyklus legen wir fest: $x_o = 1$.
Nun können wir mit der Iterationsformel aus den Größen a und x_o das neue x_1 berechnen. Die Interationsformel wird solange wiederholt, bis die Differenz zweier aufeinanderfolgender Näherungswerte kleiner als ε wird.
Als nächstes schreiben wir in den Programmablaufplan die Testanweisung $|x_1 - x_o| < \varepsilon$? Den Nein-Ausgang nehmen wir nach links. Wir setzen $x_o = x_1$ und können nach der Iterationsformel ein neues x_1 berechnen. Wir schließen den Zyklus, indem wir die Programmlinie vor der Iterationsanweisung mit der durchgehenden Programmlinie zusammenführen.
Im Gegensatz zu den vorangegangenen Beispielen wissen wir nicht, wie oft der Zyklus durchlaufen wird.
Man spricht hier von einem Iterationszyklus.

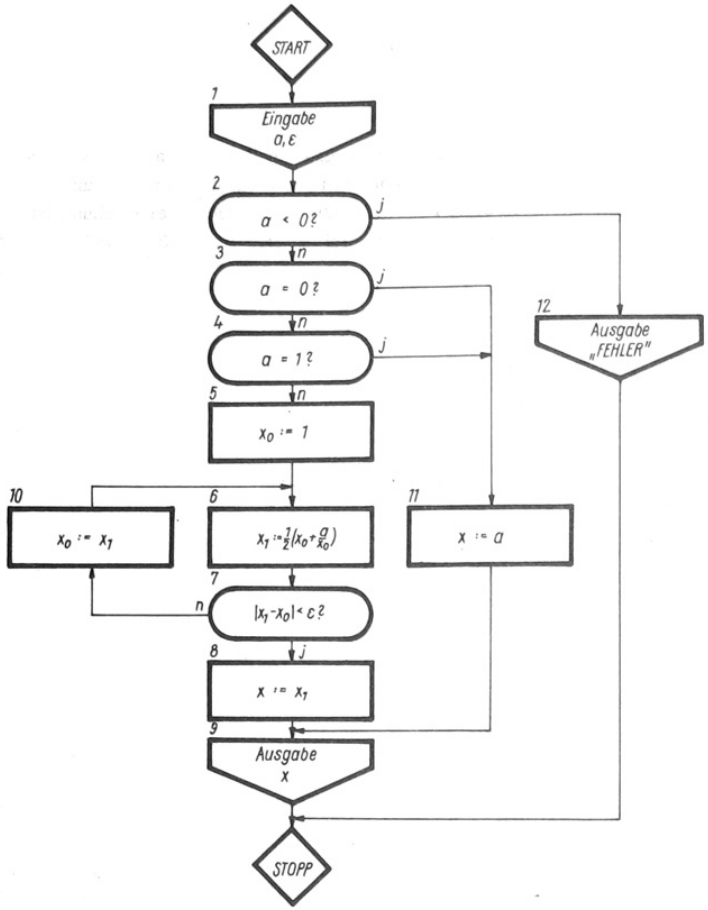


Abbildung 5
Programmablaufplan für die Berechnung der Quadratwurzel $x = \sqrt{a}$

Ein Interationszyklus wird solange durchlaufen, bis die Testanweisung mit Ja beantwortet wird.

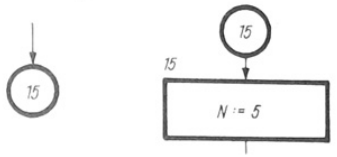
Wird die Testanweisung mit Ja beantwortet, setzen wir die gesuchte $\sqrt{x}$ **gleich dem zuletzt erhaltenen Wert** x_1, geben x aus und beenden den Programmablaufplan mit dem Stopp-Symbol.
Die beiden letzten Sinnbilder benutzen wir auch, wenn die Wurzel (aus 0 und 1) nicht nach dem Algorithmus, sondern durch die Anweisung x := a ermittelt wurde, indem wir die Programmlinie von dort diesem Teil des Ablaufs zuführen.
In dem Programmablaufplan ist noch der Zweig offen, der erreicht wird, wenn a < 0, also negativ, ist. Da wir in diesem Fall eine Wurzel nicht berechnen können, geben wir das Wort "FEHLER" aus, indem wir das Ausgabesinnbild schreiben und die Programmlinie ebenfalls zum Stopp-Symbol führen.
Anhand verschiedener Zahlenwerte kann mit dem Trockentest der Programmablaufplan leicht überprüft werden.

3. Abschließende Bemerkungen zur Aufstellung von Programmablaufplänen

3.1. Die Konnektoren

Häufig wird es nicht möglich sein, den Programmablaufplan auf einer Seite unterzubringen. In diesem Fall unterbricht man die Programmlinie und setzt sie auf der nächsten Seite fort. Das Symbol für die Unterbrechung ist der Konnektor. Man beendet die Programmlinie mit einem Kreis und beginnt auf der nächsten Seite mit dem gleichen Symbol.

In die Konnektoren wird die Nummer des folgenden Ablaufschrittes eingetragen, zum Beispiel bedeutet



Fortsetzung bei dem Konnektor mit der Nummer 15. Auf diesen Konnektor folgt das Sinnbild mit der Nummer 15.
Auch bei Testanweisungen wird der Konnektor verwendet, wenn von einem Ausgang des Testsymbols die Programmlinie zu einer Stelle geführt werden soll, die sich auf einer anderen Seite des Programmablaufplans befindet.

Man kann mit Hilfe der Konnektoren auch das Überschneiden von Programmlinien verhindern und damit die Übersichtlichkeit des Programmablaufplans erhöhen.

3.2. Methodische Hinweise

Wir haben in unseren Beispielen nur Programmablaufpläne für relativ einfache und übersichtliche Aufgaben behandelt.
Für umfangreiche Probleme wird man den Programmablaufplan in der Regel in mehreren Stufen mit wachsender Detailliertheit aufstellen. In der ersten Stufe wird man sich nicht auf Einzelheiten festlegen, sondern bestrebt sein, den Gesamtüberblick zu erfassen. Einzelne Teile dieses groben Programmablaufplans wird man dann in ihrer Darstellung verfeinern, um schließlich daraus das EDVA-Programm zu entwickeln. Wir konnten bei unseren relativ einfachen Beispielen darauf nicht im einzelnen eingehen.
Bei Erarbeitung eines Programmablaufplans wird man oft feststellen, daß man weiter oben noch bestimmte Anweisungen einfügen muß – beispielsweise zur Herstellung der Anfangsbedingungen für einen Zyklus.
Man beginnt deshalb die Aufstellung des Programmablaufplans vielfach mit dem Kern der Aufgabe – etwa dem Zyklus im Beispiel der Quadratwurzel-

berechnung (Anweisungen 6, 7, 10 in Abbildung 5) – und schreibt, davon ausgehend, die anderen Anweisungen, auch die im Programmablaufplan noch vorher auftretenden (Anweisungen 1 bis 5 in Abbildung 5) zur Schaffung der Voraussetzungen für den Zyklus.

Jeder Programmablaufplan sollte durch einen Trockentest auf seine Richtigkeit überprüft werden, indem man den Ablauf an verschiedenen Zahlenbeispielen verfolgt. Diese sollten so gewählt werden, daß möglichst alle Zweige des Programmablaufplans wenigstens einmal durchlaufen werden.

Detaillierte Programmablaufpläne sollten stets von Programmierern aufgestellt werden, die den Programmablaufplan als Grundlage für das aufzustellende Programm benutzen. Je weiter man einen Programmablaufplan verfeinert, um so mehr müssen die Besonderheiten der zur Verfügung stehenden Anlage beachtet werden, um deren Vorteile auch ausnutzen zu können. Von Organisatoren, die die Programmierung selbst nicht beherrschen, sollten deshalb nur relativ grobe problemorientierte Programmablaufpläne aufgestellt werden. Hier können auch Anweisungen wie

$$U := \sum_{i=1}^{n} P_i A_i$$

in geschlossener Form erscheinen, ohne daß angegeben wird, wie die Produktsumme im einzelnen zu errechnen ist. Das wird jeder Programmierer ohne Programmablaufplan ausführen können.

3.3. Die Leitlinienmethode

Bei der Leitlinienmethode handelt es sich um eine vereinfachte Darstellung von Programmabläufen. Es werden weniger Symbole als bei der Kästchenmethode verwandt, wodurch das ständige Zeichnen der Kästchen wegfällt.

Kurze waagerechte, die Programmlinie kreuzende Striche kennzeichnen die Zuordnung der Operationen, deren mathematische oder textliche Erläuterungen rechts daneben gesetzt werden.

Wesentlich anders wird in der Leitlinienmethode die Testanweisung dargestellt. Man schreibt als Abkürzung für "Prüfe" P und setzt die Prüfbedingung in geschweifte Klammern. Der Ja-Zweig wird durch P, der Nein-Zweig durch $\overline{P}$ gekennzeichnet.

In der Abbildung 6 ist als Beispiel der Programmablaufplan der Quadratwurzelberechnung in Leitlinienmethode angegeben. Nach der ausführlichen Behandlung der Kästchenmethode dürfte es nicht schwerfallen, auch einen in der Leitlinienmethode aufgestellten Programmablaufplan zu lesen und zu verstehen.

Der Vorteil der Leitlinienmethode gegenüber der Kästchenmethode besteht im geringeren Zeichenaufwand. Andererseits ist ein Programmablaufplan in der Leitlinienmethode nicht so übersichtlich wie in der Kästchenmethode. In der Praxis wird deshalb für das Konzept eines Programmablaufplans häufig die Leitlinienmethode gewählt, während für endgültige Darstellungen und Dokumentationen die Kästchenmethode verwendet wird.

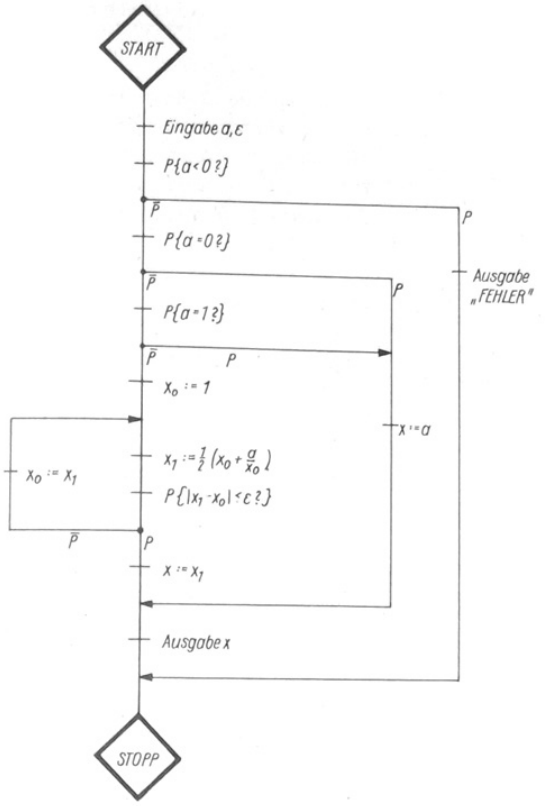
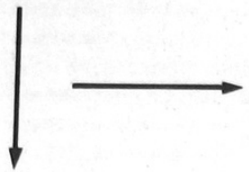
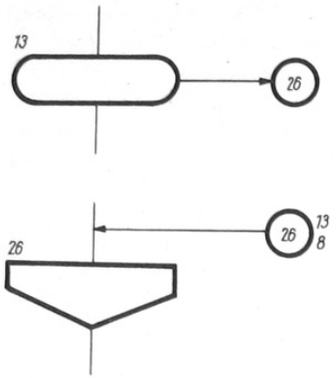
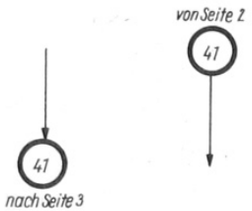
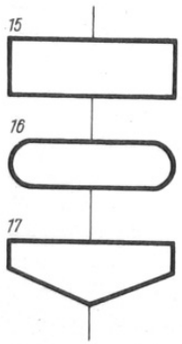
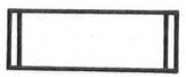
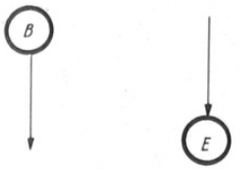
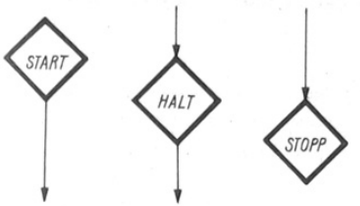


Abbildung 6
Programmablaufplan für die Quadratwurzelberechnung (Leitlinienmethode)

4. Die Zusammenstellung der wesentlichen Sinnbilder für Programmablaufpläne (Kästchenmethode)

Sinnbild	Bedeutung für Programmablaufpläne
	Ein- und Ausgabeoperation
	Operation, allgemein (insbesondere für Ergibtanweisungen)
	Verzweigen des Programms (Vergleichsoperation)

Sinnbild	Bedeutung für Programmablaufpläne
	Programmlinie (Übergang zum nächsten Ablaufschritt. Vorzugsrichtung von oben nach unten, von links nach rechts. Falls unmißverständlich, kann Pfeilspitze weggelassen werden.)
	Zeichen-Konnektor (Für Unterbrechung und Fortsetzung der Programmlinie aus zeichnerischen Gründen. Zusammengehörige Konnektoren erhalten die gleiche Bezeichnung, die sich nach der Nummer des folgenden Programmschrittes richtet, z.B. 26. Es können mehrere Unterbrechungsstellen zu einer Einsprungstelle führen. Nummern neben diesen kennzeichnen ggf. den jeweils letzten Schritt vor der Unterbrechung, z.B. 13, 8.)
	Seiten-Konnektor (Für Unterbrechung und Fortsetzung der Programmlinie auf einer anderen Seite, z.B. Fortsetzung bei Konnektor 41 auf Seite 3)
	Fortlaufende Numerierung der Programmschritte (links über den Sinnbildern, z.B. 15, 16, 17)
	Unterprogramm-Block, Programmteil

Sinnbild	Bedeutung für Programmablaufpläne
	Beginn und Ende eines Unterprogramms, Programmteils (Für Detaillierung von Unterprogrammen, Programmteilen außerhalb der Programmhauptlinie)
	Programmanfang und -ende (START, STOPP oder HALT – HALT bei Zwischenhalt – sind in das Sinnbild einzuschreiben.)

Hinweis

für die Direktoren
der prüfungsberechtigten
Institutionen

Zur organisatorischen Vorbereitung der Prüfungen und der Belieferung mit Prüfungsunterlagen benötigt die Fernsehakademie von allen prüfungsberechtigten Institutionen die Anzahl der voraussichtlich an den Prüfungen teilnehmenden Fernsehzuschauer.
Diese Angabe ist bis zum 30. 9. 1969 auf einer Postkarte oder über Fernschreiber (Telex 011432) mit genauer Bezeichnung des Absenders an folgende Adresse einzusenden:

Deutscher Fernsehfunk
Fernsehakademie
Redaktion EDV

1199 Berlin
Rudower Chaussee

Anmeldung

für die Prüfungen
zum Fernsehkurs
"EDV", Teil 1
Termin: 15. 9. 1969

An die Volkshochschule/Betriebsakademie

Ich möchte mich zur Teilnahme an der Prüfung zum Fernsehkurs "Elektronische Datenverarbeitung (Teil I)" anmelden.

Name Vorname geb. am

Wohnort

Straße

Tätigkeit

Betrieb/Institution

Unterschrift

Einige Fragen

zur Vorbereitung auf die
Prüfung zum Fernsehkurs
"Elektronische
Datenverarbeitung"

Wiederholungsfragen zum Themenkomplex II
"Mathematische und technische Grundlagen"

1. Welche Bauelemente zur Informationsdarstellung haben bisher in der Datenverarbeitung Verwendung gefunden?
2. Wieviel verschiedene Ziffern hat das Dualsystem?
3. Nennen Sie die Basen des Dezimal- und des Dualsystems!
4. Übersetzen Sie folgende Tetraden ins Dezimalsystem:
   L000, L00L, 00LL (Normalverschlüsselung)
5. Schreiben Sie folgende Dezimalzahlen als Dualtetraden:
   2, 4, 7
6. Enthält ein Lochband bei der Verschlüsselung von Zahlen Dezimalzeichen oder Binärzeichen?
7. Nennen Sie 3 logische Grundoperationen!
8. Wie heißen die Schaltungen, mit denen die 3 logischen Grundoperationen realisiert werden?
9. Was versteht man unter einem Speicherelement?
10. Wieviel Bits sind zur Speicherung einer Dualtetrade notwendig?
11. Haben kombinatorische Schaltungen speichernde Eigenschaften?
12. Nennen Sie je ein Beispiel für einen Festspeicher und einen Arbeitsspeicher!
13. Nennen Sie 2 Möglichkeiten der Kommadarstellung in Rechenwerken!
14. Nennen Sie 2 Möglichkeiten der Wortdarstellung in Rechenwerken!
15. Was versteht man unter serieller, paralleler bzw. serienparalleler Arbeitsweise einer Datenverarbeitungsanlage?
16. Zeichnen Sie das Blockschaltbild eines Addiators für 2 Zahlen mit Berücksichtigung eines Übertrags (serielle Arbeitsweise)!

Matrizenmodelle

FÜR DIE PLANUNG
IN DER METALLVERARBEITENDEN
INDUSTRIE

von Dr. Hellmut Seidel

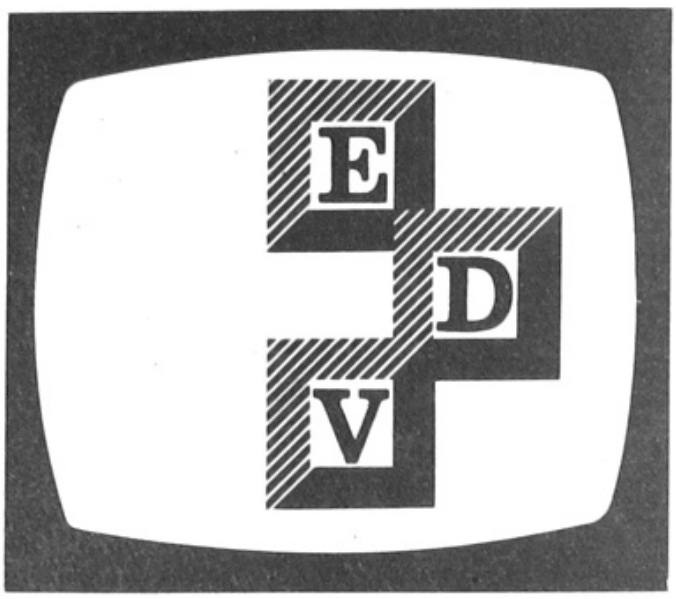
Schriftenreihe "Datenverarbeitung"
Herausgeber: Institut für Datenverarbeitung,
Dresden

264 Seiten · 19 Abbildungen · 6 Beilagen
Broschur · 20,40 M

An Hand von konkreten Beispielen werden Matrizenmodelle entwickelt und ihre Anwendungsmöglichkeiten für die Planung eingehend erläutert. Die dargestellten Modelle wurden bereits in der Praxis erprobt und von den Experimentierbetrieben in die Planung eingeführt. Diese Modelle sind direkt bzw. leicht abgewandelt auch in anderen Industriezweigen anwendbar.

VERLAG
DIE WIRTSCHAFT
BERLIN

QUALIFIZIERUNG FREI HAUS



Das Programm des Deutschen Fernsehfunks ist vielseitig, es bietet auch Qualifizierung zu Hause am Bildschirm: Der Fernsehkurs "Elektronische Datenverarbeitung" hilft Ihnen, Ihre beruflichen Aufgaben von morgen zu meistern. Der Dozent kommt ins Haus – bequemer geht es nicht! Meinen Sie nicht auch, daß sich die Anschaffung eines Fernsehgerätes lohnt?

Unsere Fachfilialen informieren Sie gern über die günstigen Teilzahlungsbedingungen.

VEB INDUSTRIEVERTRIEB RUNDFUNK UND FERNSEHEN

RFT **RADIO-*television***

Wissensspeicher Grundlagen der Datenverarbeitung

160 Seiten · 10 Abb. · 59 graf. Darstellungen · Broschur · 5,-M

Mit der Einführung der neuen Grundlagenfächer Elektronik, BMSR-Technik und Datenverarbeitung in der sozialistischen Berufsausbildung wird den jungen Menschen ein praktisches und theoretisches Rüstzeug über Automatisierungstechnik und die Datenverarbeitung vermittelt, um sie zu aktiven Beherrschern und Mitgestaltern der modernen Produktion zu entwickeln und ihnen den Reichtum und die Fülle der Aufgaben zur Durchsetzung der wissenschaftlich-technischen Revolution in der sozialistischen Gesellschaft geistig zu erschließen. Diese neue Form der berufsbildenden Literatur soll vorwiegend die Funktion des Speicherns von Wissen übernehmen, das Gedächtnis entlasten und das Erlernte unter den verschiedensten Bedingungen schnell und sicher reaktivieren.

VERLAG DIE WIRTSCHAFT BERLIN